Compilers

Dr. Sherin ElGokhy

Lecture#9

SLR Parsing

Outline

- Recognizing Viable Prefixes
- SLR parsing

An NFA Recognizing Viable Prefixes

1. Add a dummy production $S' \rightarrow S$ to G
2. The NFA states are the items of G
   - Including the extra production
3. For item $E \rightarrow \alpha.X\beta$ add transition

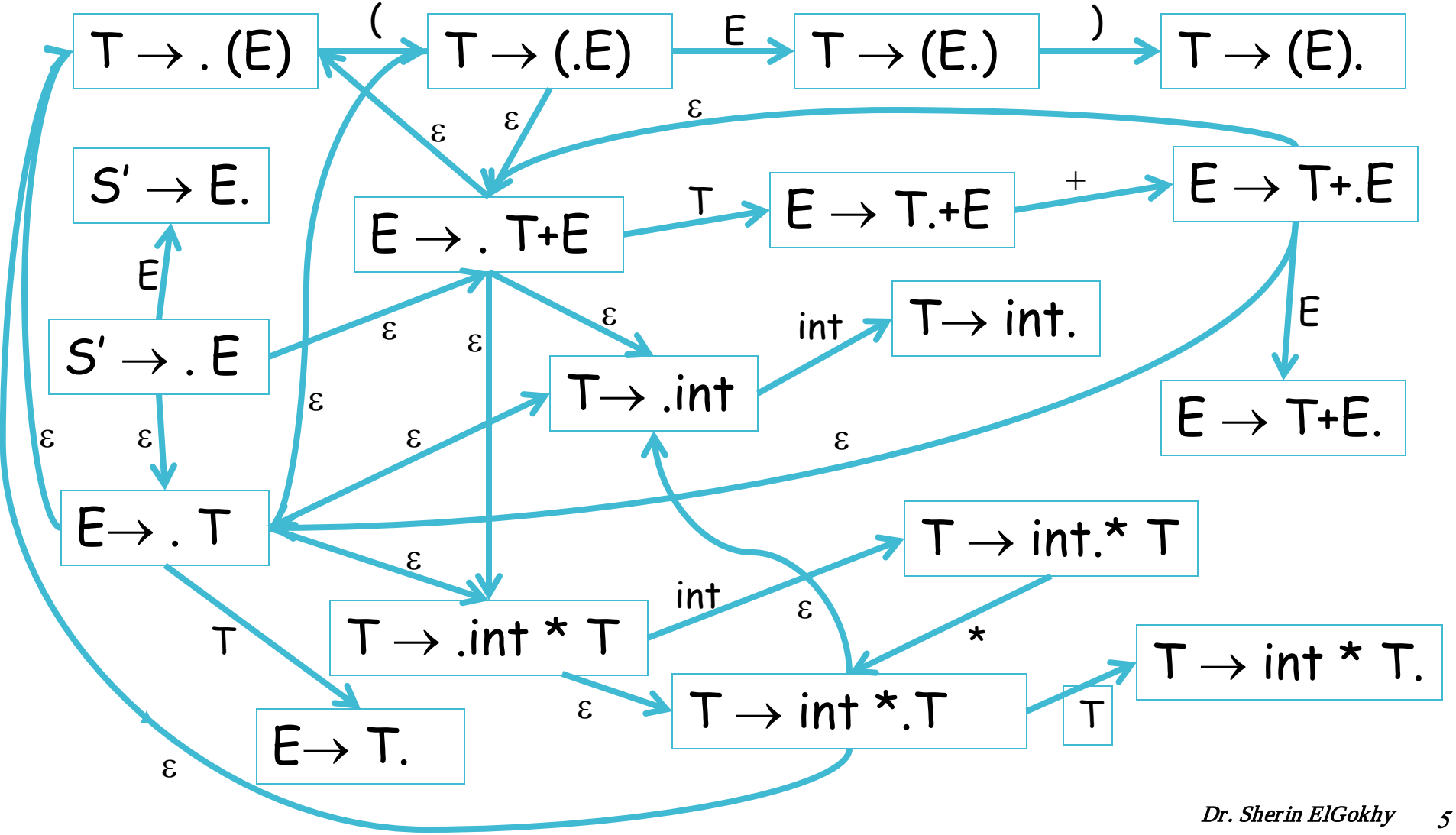
   $$E \rightarrow \alpha.X\beta \rightarrow^{X} E \rightarrow \alpha X.\beta$$

4. For item $E \rightarrow \alpha.X\beta$ and production $X \rightarrow \gamma$ add

   $$E \rightarrow \alpha.X\beta \rightarrow^{\varepsilon} X \rightarrow .\gamma$$

5. Every state is an accepting state
6. Start state is $S' \rightarrow .S$

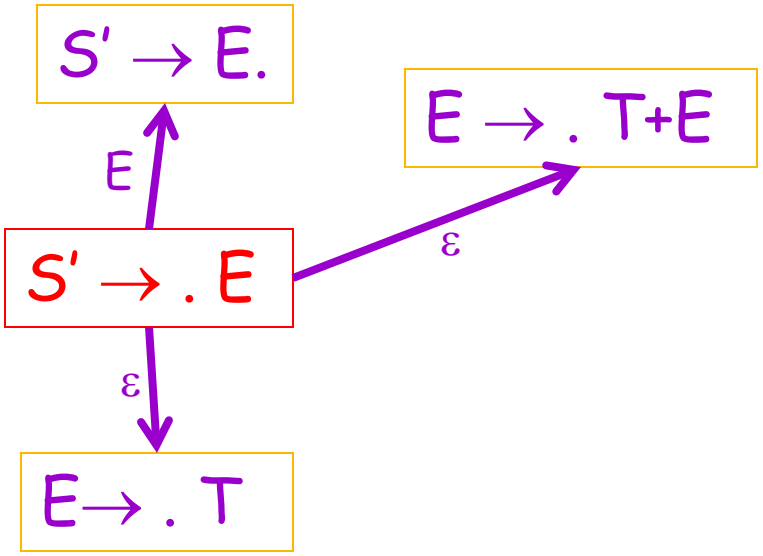
NFA for Viable Prefixes of the Example



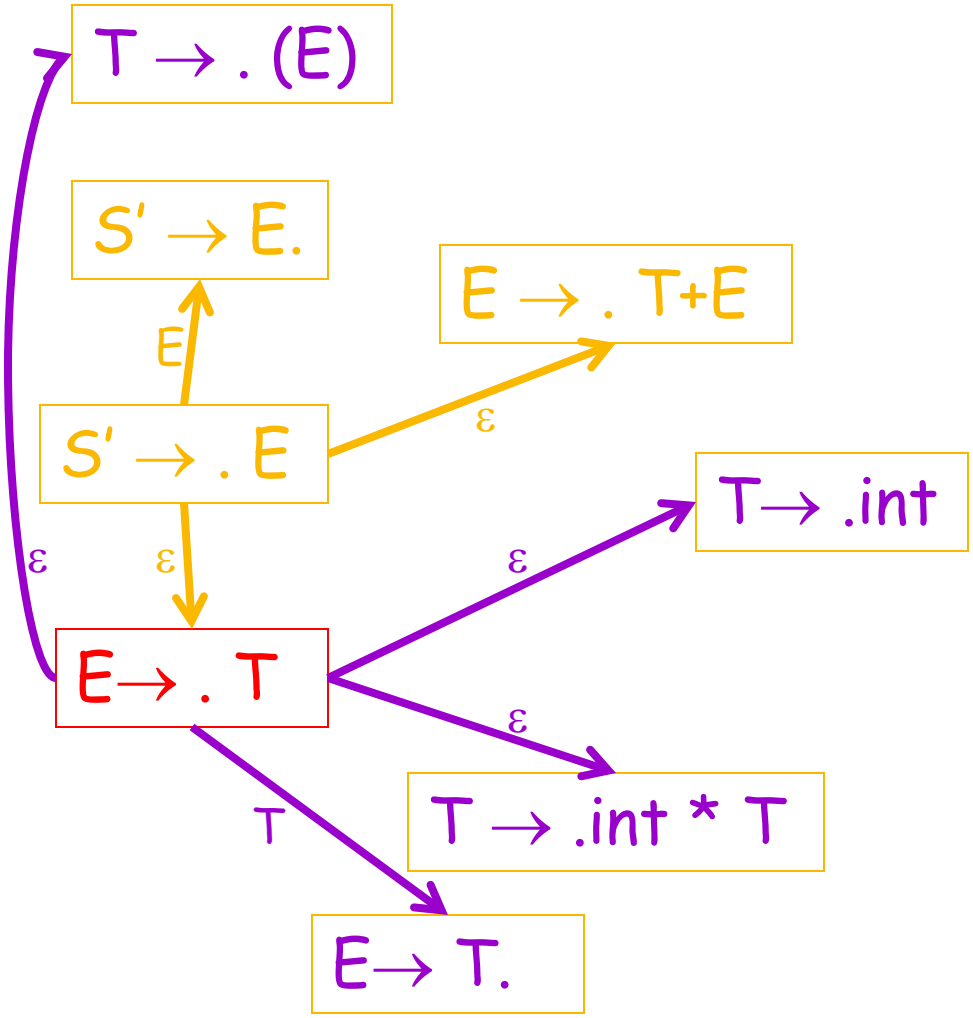
NFA for Viable Prefixes of the Example Detail (1)

$S' \rightarrow . E$

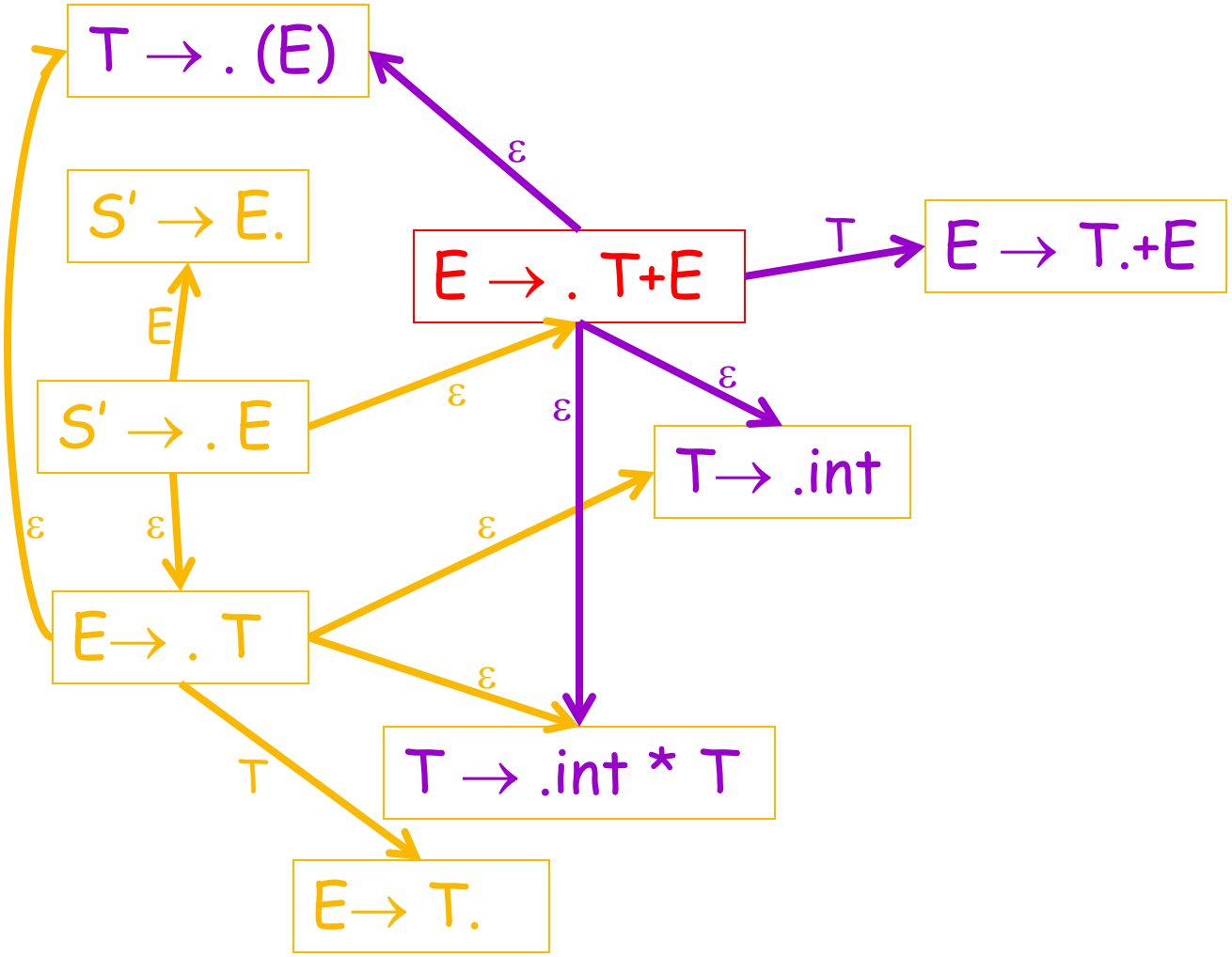
NFA for Viable Prefixes of the Example Detail (2)



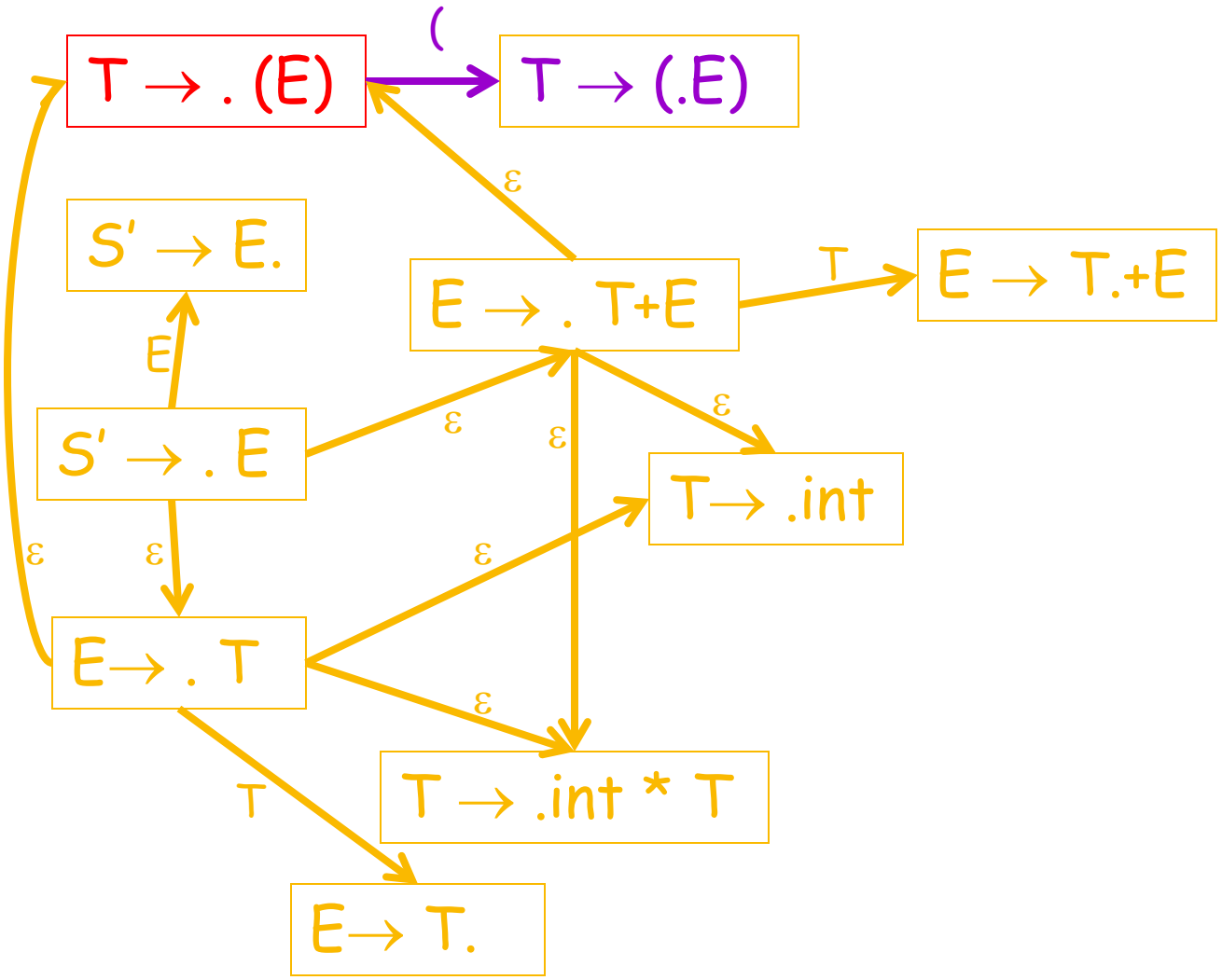
NFA for Viable Prefixes of the Example Detail (3)



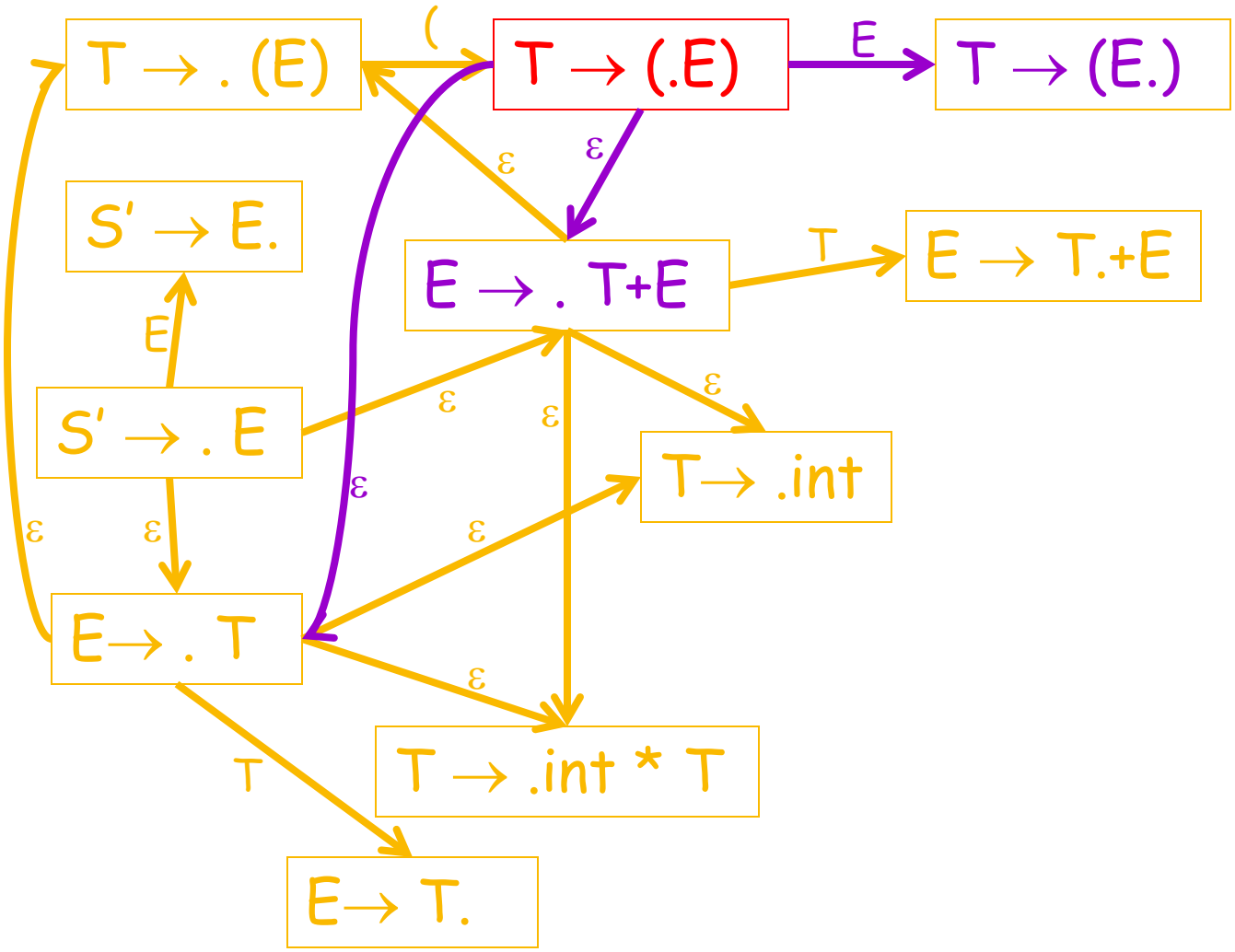
NFA for Viable Prefixes of the Example Detail (4)



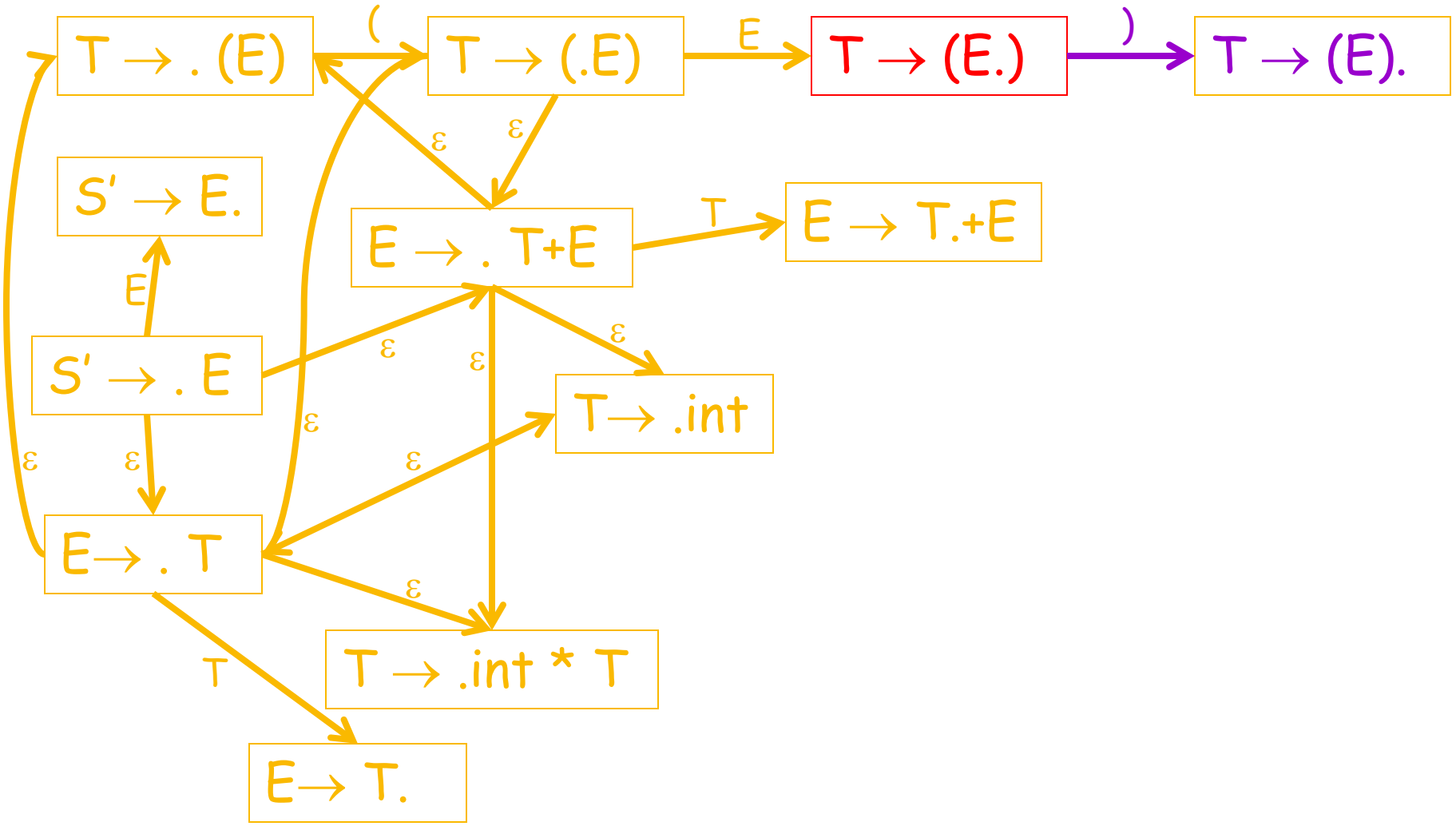
NFA for Viable Prefixes of the Example Detail (5)



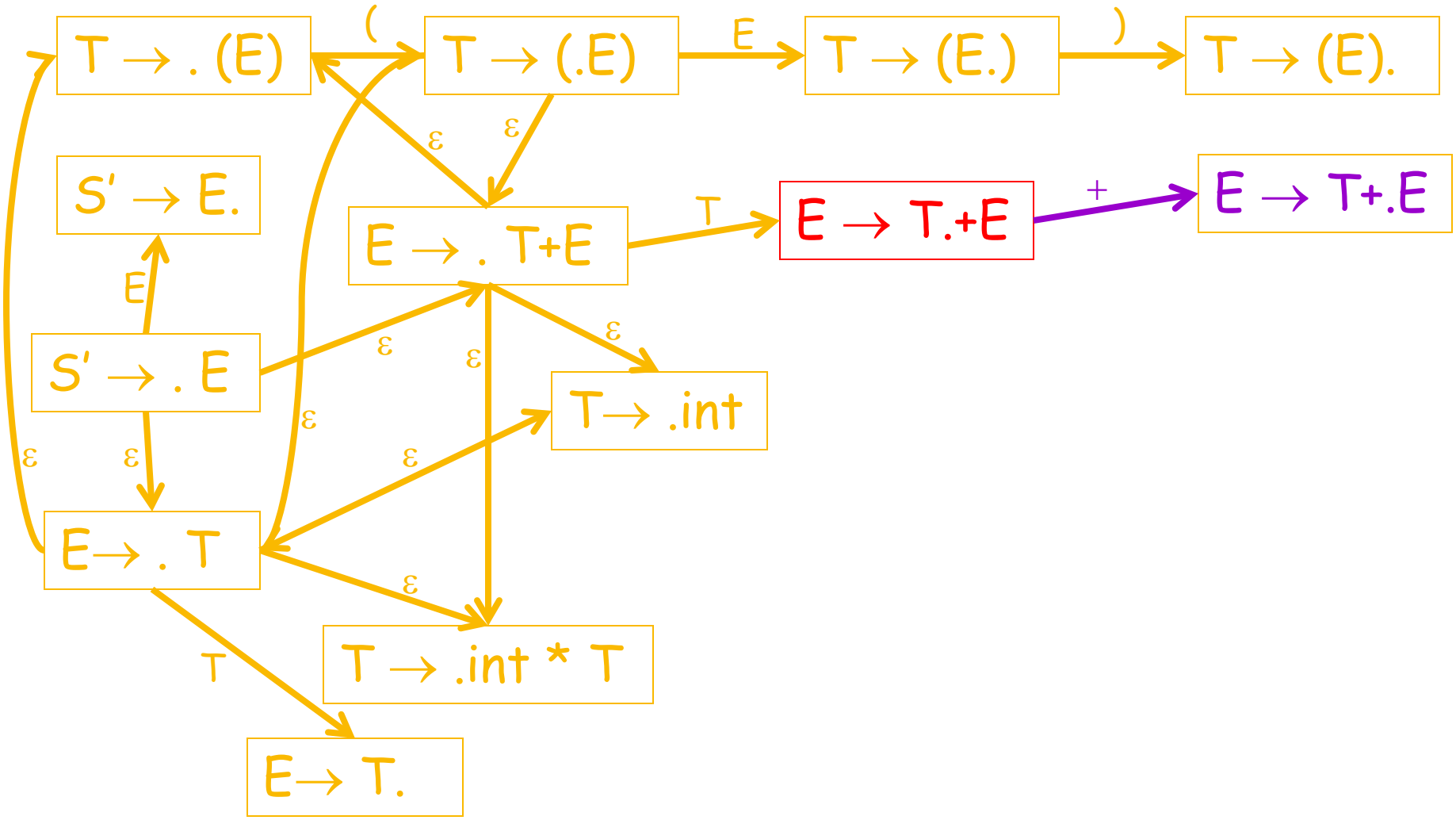
NFA for Viable Prefixes of the Example Detail (6)



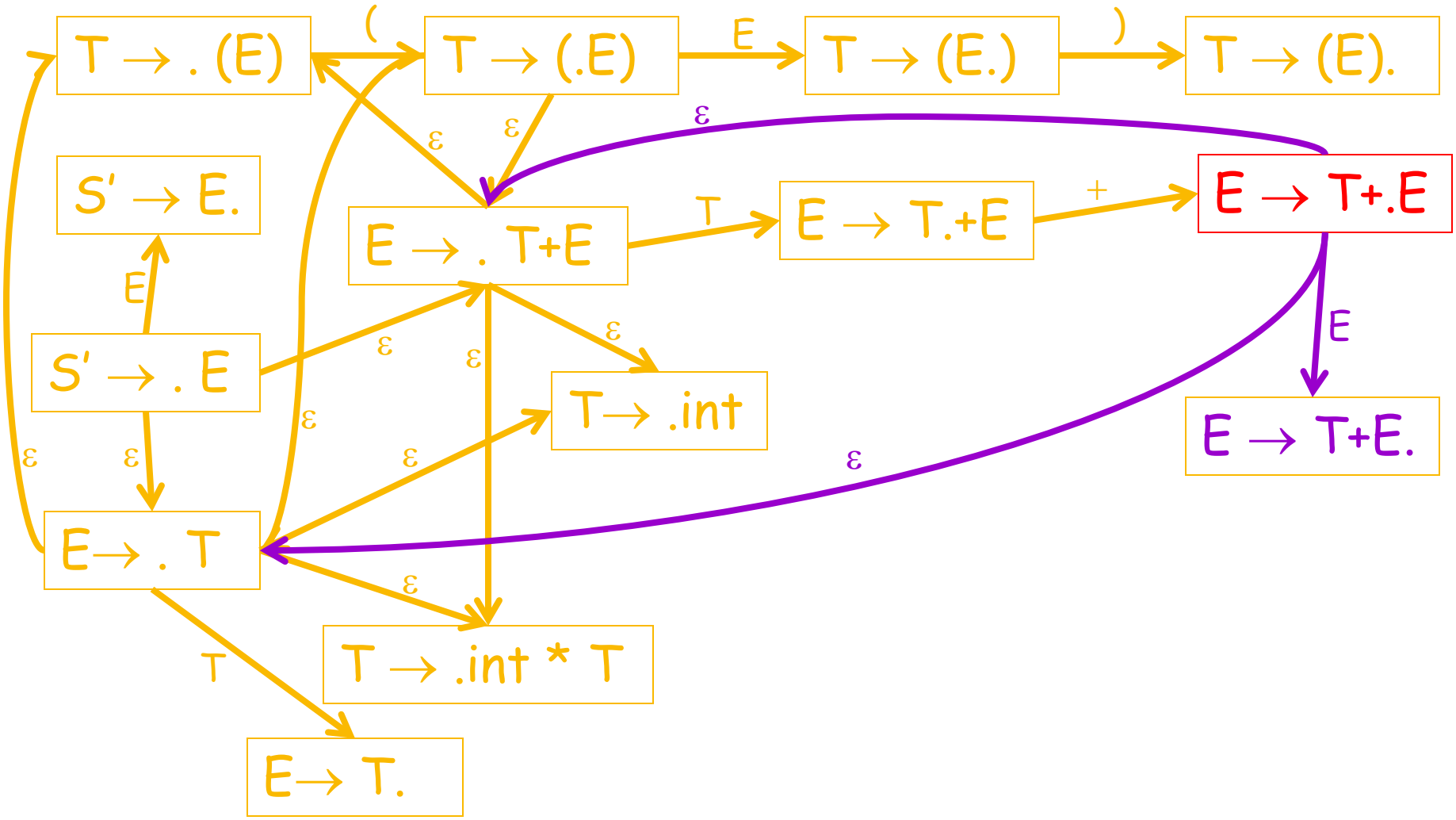
NFA for Viable Prefixes of the Example Detail (7)



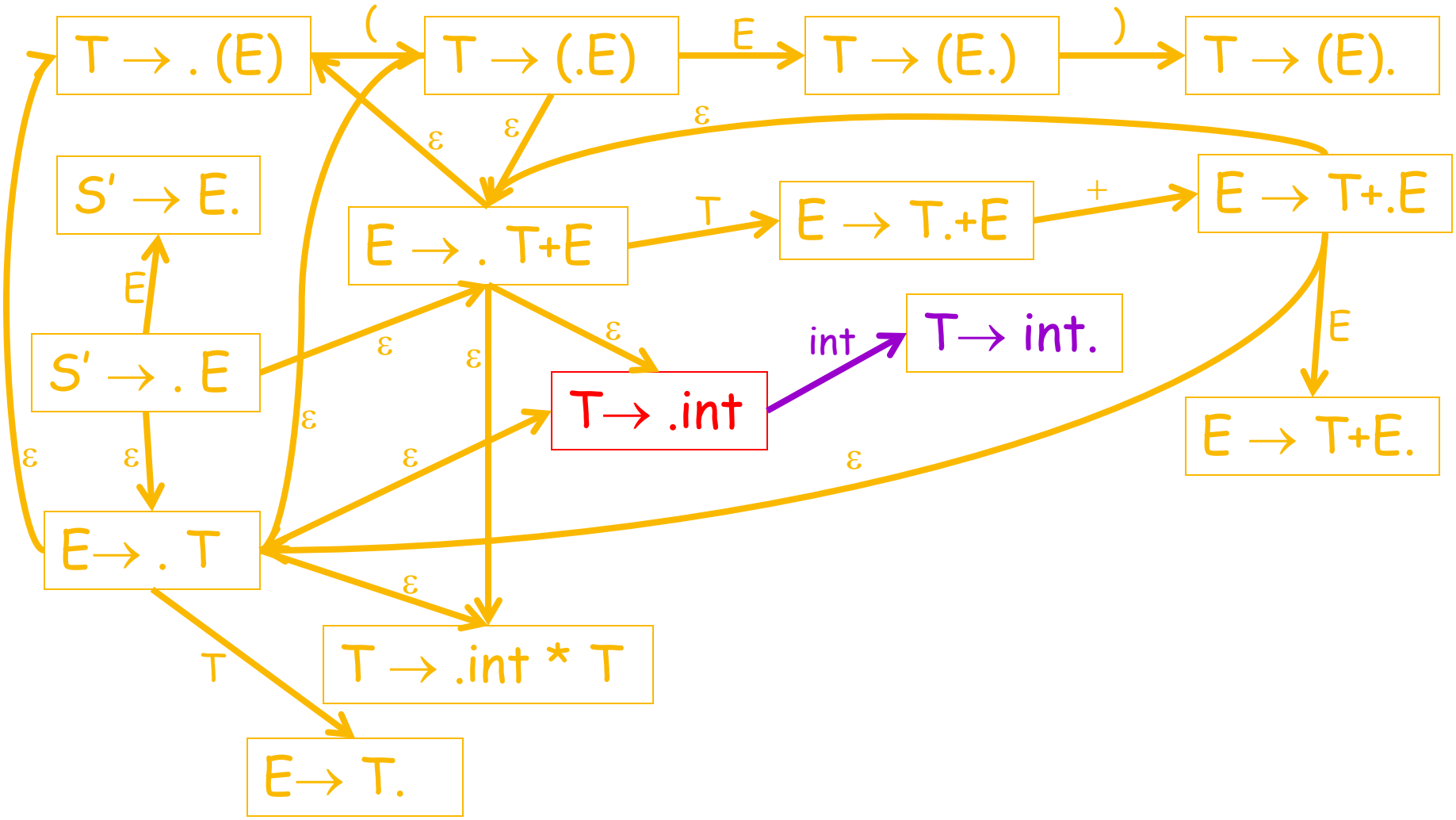
NFA for Viable Prefixes of the Example Detail (8)



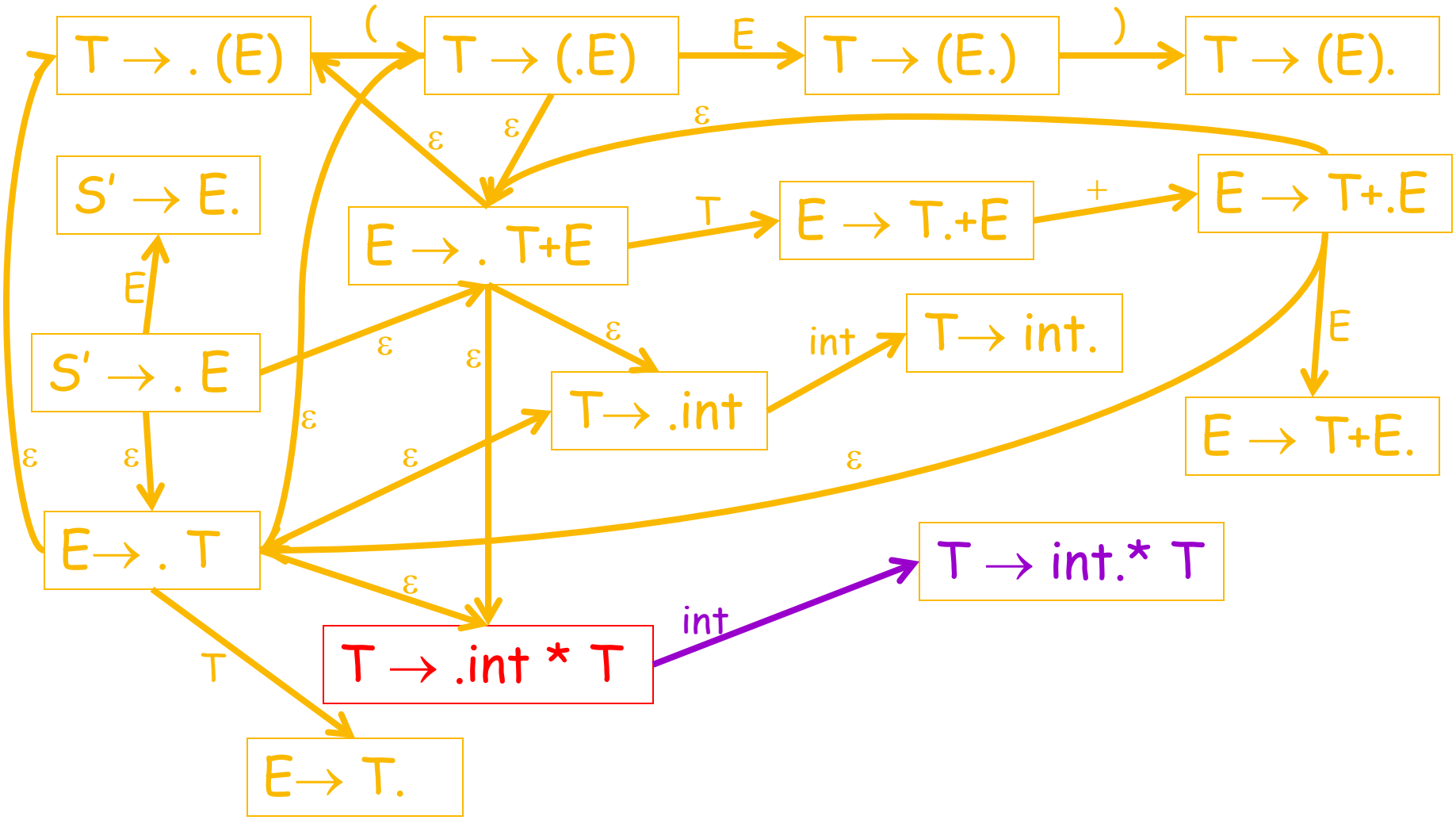
NFA for Viable Prefixes of the Example Detail (9)



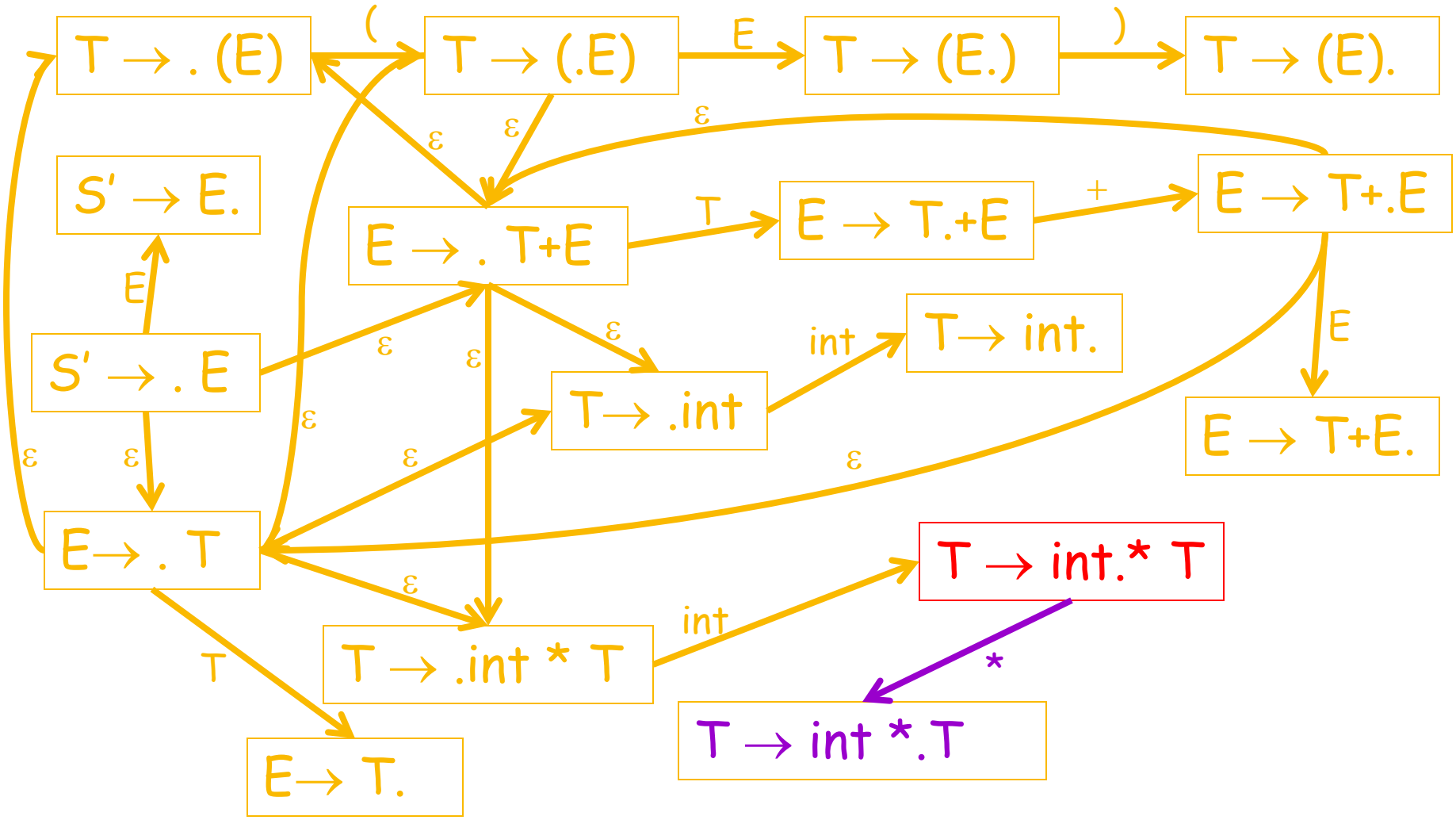
NFA for Viable Prefixes of the Example Detail (10)



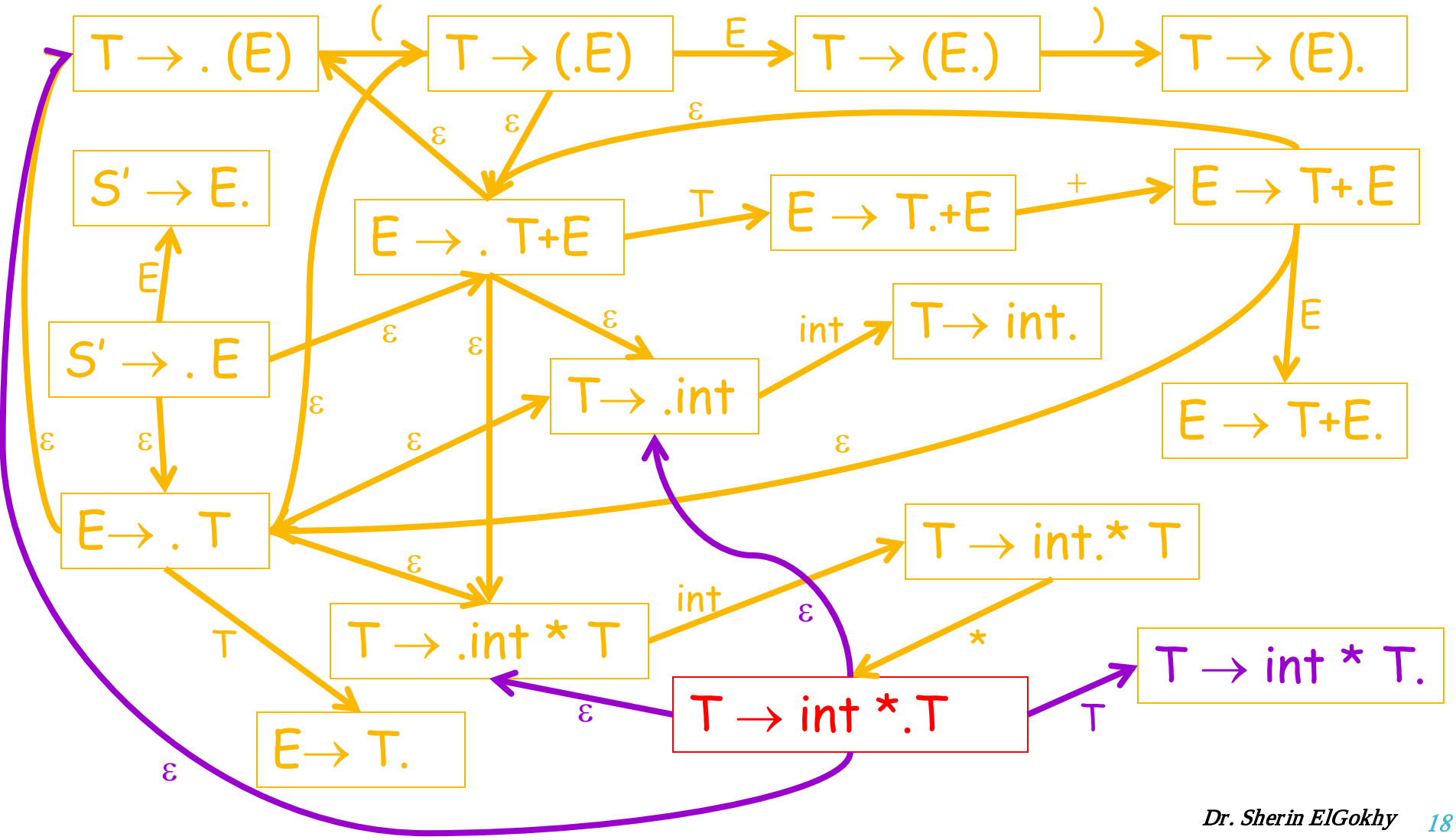
NFA for Viable Prefixes of the Example Detail (11)



NFA for Viable Prefixes of the Example Detail (12)



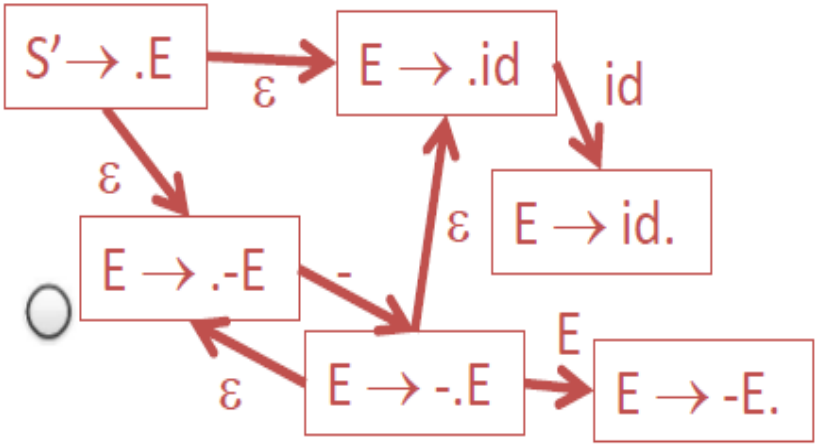
NFA for Viable Prefixes of the Example Detail (13)

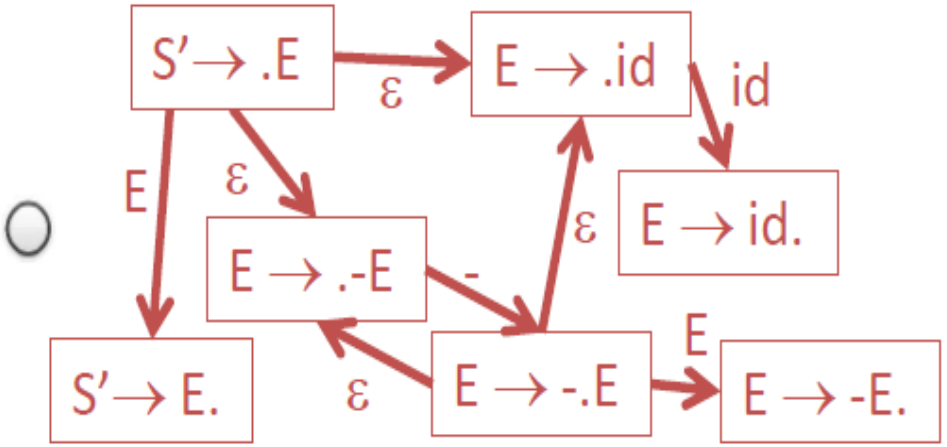


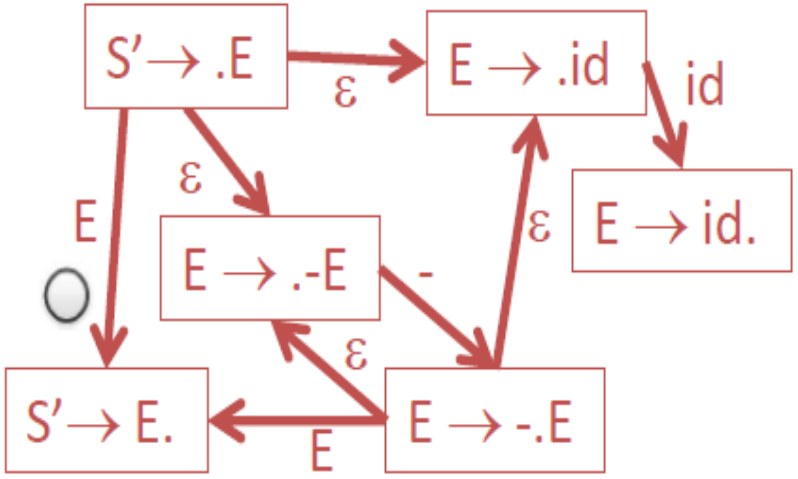
Quiz

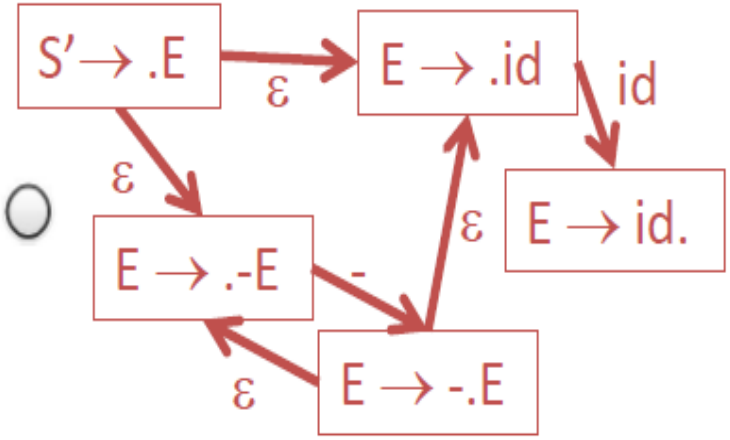
Choose the correct NFA for the given grammar

S′→ E E → -E | id

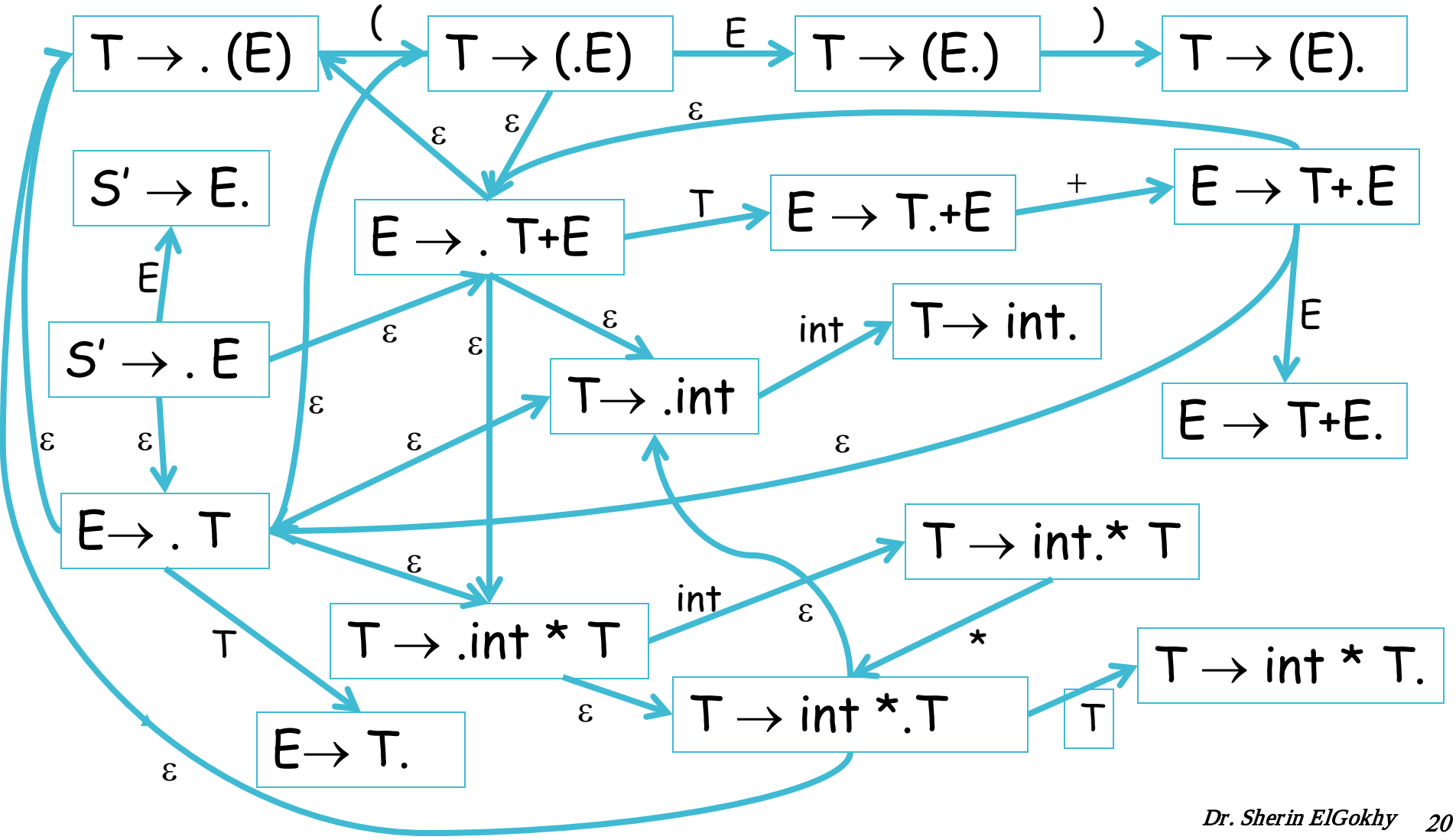




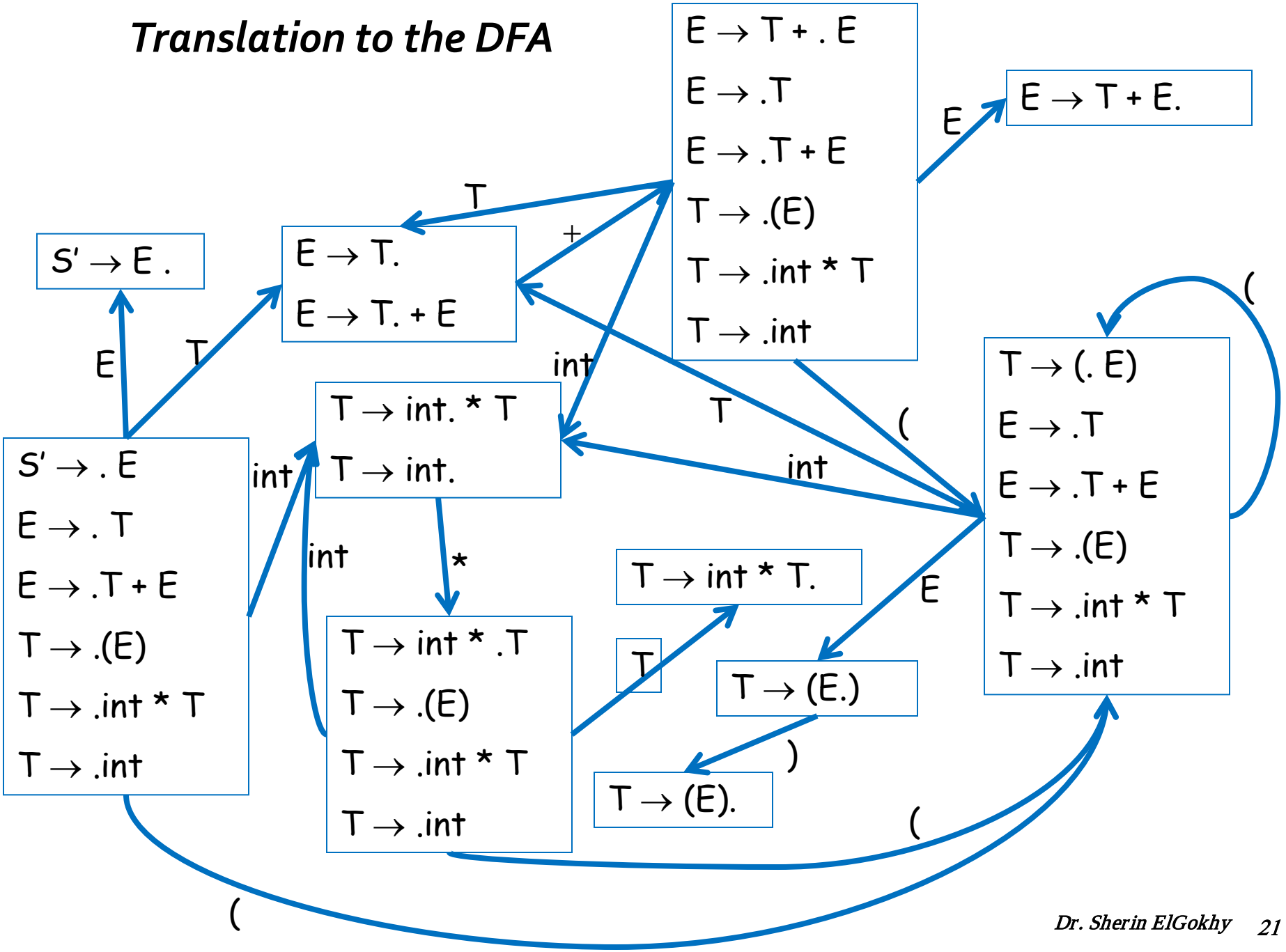




A complete NFA for recognizing Viable Prefixes of the considered Example



Translation to the DFA



DFA

The states of the DFA are

"canonical collections of items"

or

"canonical collections of LR(0) items"

Valid Items

Item $X \rightarrow \beta.\gamma$ is *valid* for a viable prefix $\alpha\beta$ if

$$S' \rightarrow^* \alpha X \omega \rightarrow \alpha\beta\gamma\omega$$

by a series of right-most derivation steps

After parsing $\alpha\beta$, the valid items are the possible tops of the stack of items

Valid Items Example

- An item is often valid for many prefixes
- Example: The item $T \rightarrow (.E)$ is valid for prefixes

(

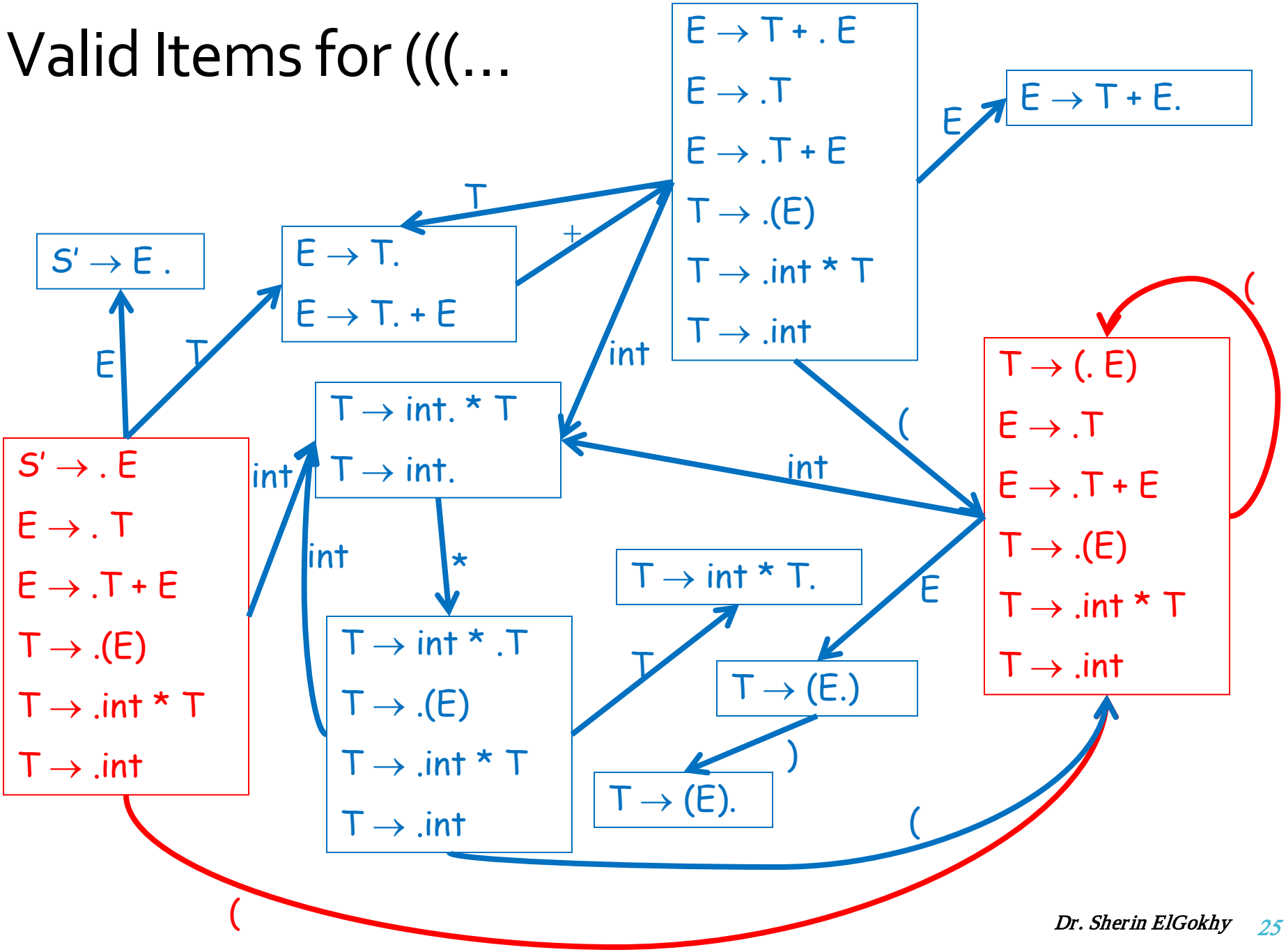
((

(((

((((

. . .

Valid Items for (((…



Quiz

Using the automaton on the previous slide, choose the valid items for the prefix: (int *

- ☐ $E \rightarrow (.E)$
- ☐ $T \rightarrow int *.T$
- ☐ $E \rightarrow .T + E$
- ☐ $T \rightarrow .int$

LR(0) Parsing

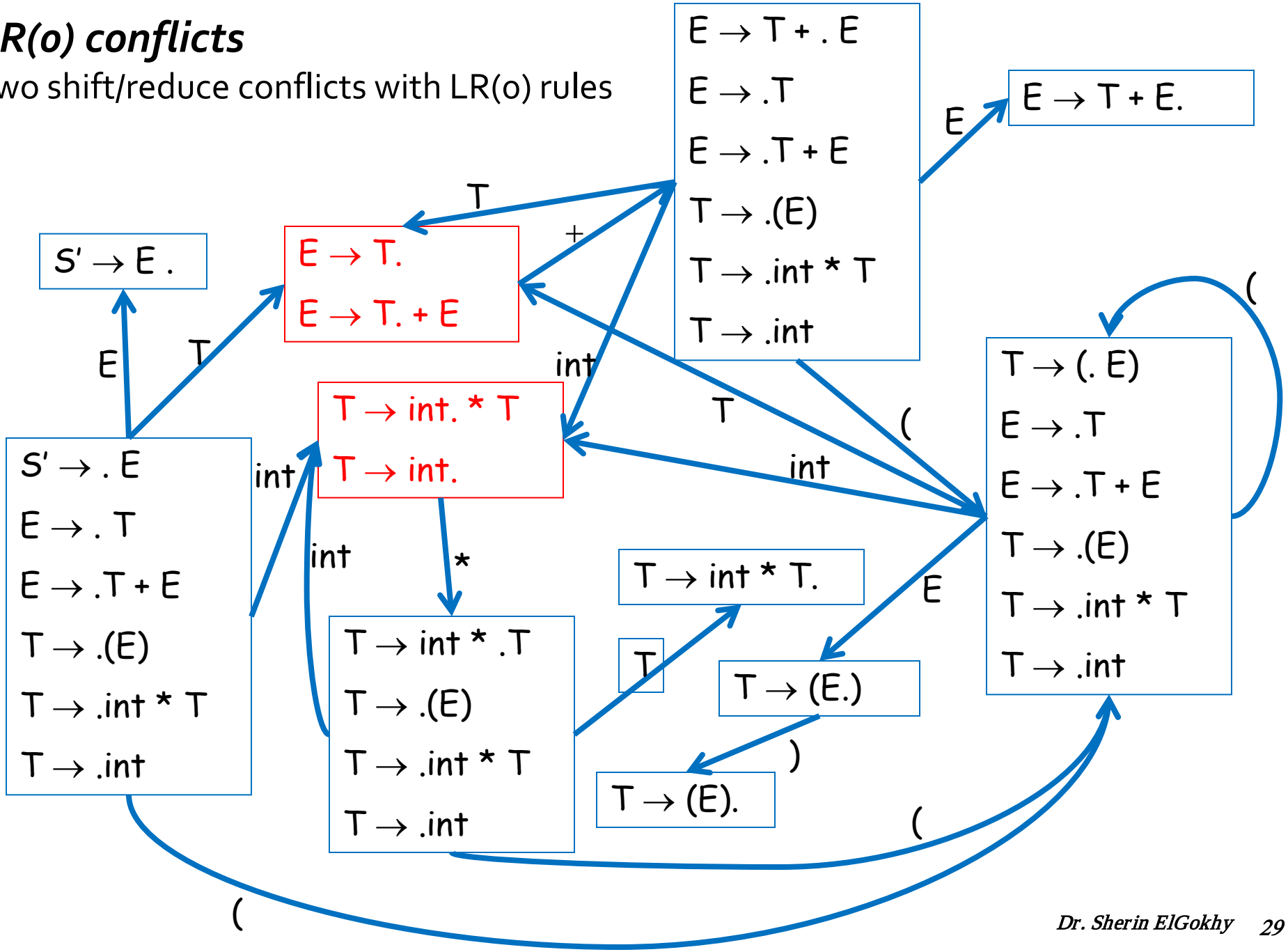
- Idea: Assume
  - stack contains α
  - next input is t
  - DFA on input α terminates in state s
- Reduce by $X \rightarrow \beta$ if
  - s contains item $X \rightarrow \beta.$
- Shift if
  - s contains item $X \rightarrow \beta.t\omega$
  - equivalent to saying s has a transition labeled t

LR(0) Conflicts

- LR(0) has a reduce/reduce conflict if:
  - Any state has two reduce items:
  - $X \rightarrow \beta.$ and $Y \rightarrow \omega.$
- LR(0) has a shift/reduce conflict if:
  - Any state has a reduce item and a shift item:
  - $X \rightarrow \beta.$ and $Y \rightarrow \omega.t\delta$

LR(0) conflicts

Two shift/reduce conflicts with LR(0) rules



SLR

- LR = "Left-to-right scan"
- SLR = "Simple LR"

- SLR improves on LR(0) shift/reduce heuristics
  - Fewer states have conflicts

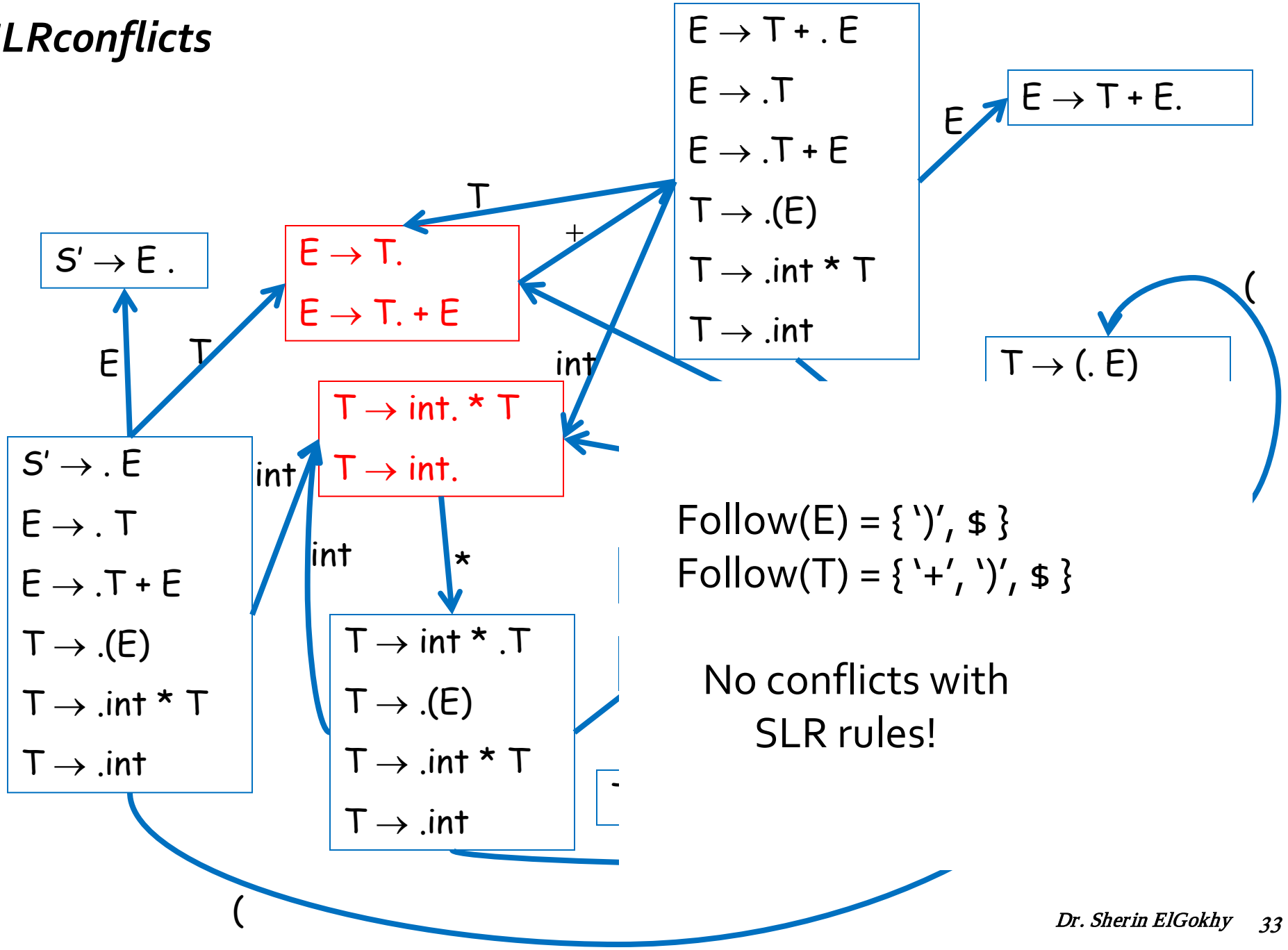
SLR Parsing

- Idea: Assume
  - stack contains α
  - next input is t
  - DFA on input α terminates in state s
- Reduce by $X \rightarrow \beta$ if
  - s contains item $X \rightarrow \beta.$
  - $t \in Follow(X)$ 
- Shift if
  - s contains item $X \rightarrow \beta.t\omega$
  - equivalent to saying s has a transition labeled t

SLR Parsing (Cont.)

- If there are conflicts under these rules, the grammar is not SLR
- The rules amount to a heuristic for detecting handles
  - The SLR grammars are those where the heuristics detect exactly the handles

SLRconflicts



Follow(E) = { ')', $ }
Follow(T) = { '+', ')', $ }

No conflicts with SLR rules!

Precedence Declarations Digression

- Lots of grammars aren't SLR
  - including all ambiguous grammars
- We can parse more grammars by using precedence declarations
  - Instructions for resolving conflicts

Precedence Declarations (Cont.)

- Consider our favorite ambiguous grammar:
  - $E \rightarrow E + E \mid E * E \mid (E) \mid int$
- The DFA for this grammar contains a state with the following items:
  - $E \rightarrow E * E .$ $E \rightarrow E . + E$
  - shift/reduce conflict!
- Declaring "* has higher precedence than +" resolves this conflict in favor of reducing

Precedence Declarations (Cont.)

- The term "precedence declaration" is misleading
- These declarations do not define precedence; they define conflict resolutions
  - Not quite the same thing!

Naïve SLR Parsing Algorithm

1. Let M be DFA for viable prefixes of G
2. Let $|x_1 \ldots x_n\$$ be initial configuration
3. Repeat until configuration is $S|\$$
   - Let $\alpha|\omega$ be current configuration
   - Run M on current stack α
   - If M rejects α, report parsing error
     - Stack α is not a viable prefix
   - If M accepts α with items I, let a be next input
     - Shift if $X \rightarrow \beta.\ a\ \gamma \in I$
     - Reduce if $X \rightarrow \beta.\ \in I$ and $a \in Follow(X)$
     - Report parsing error if neither applies

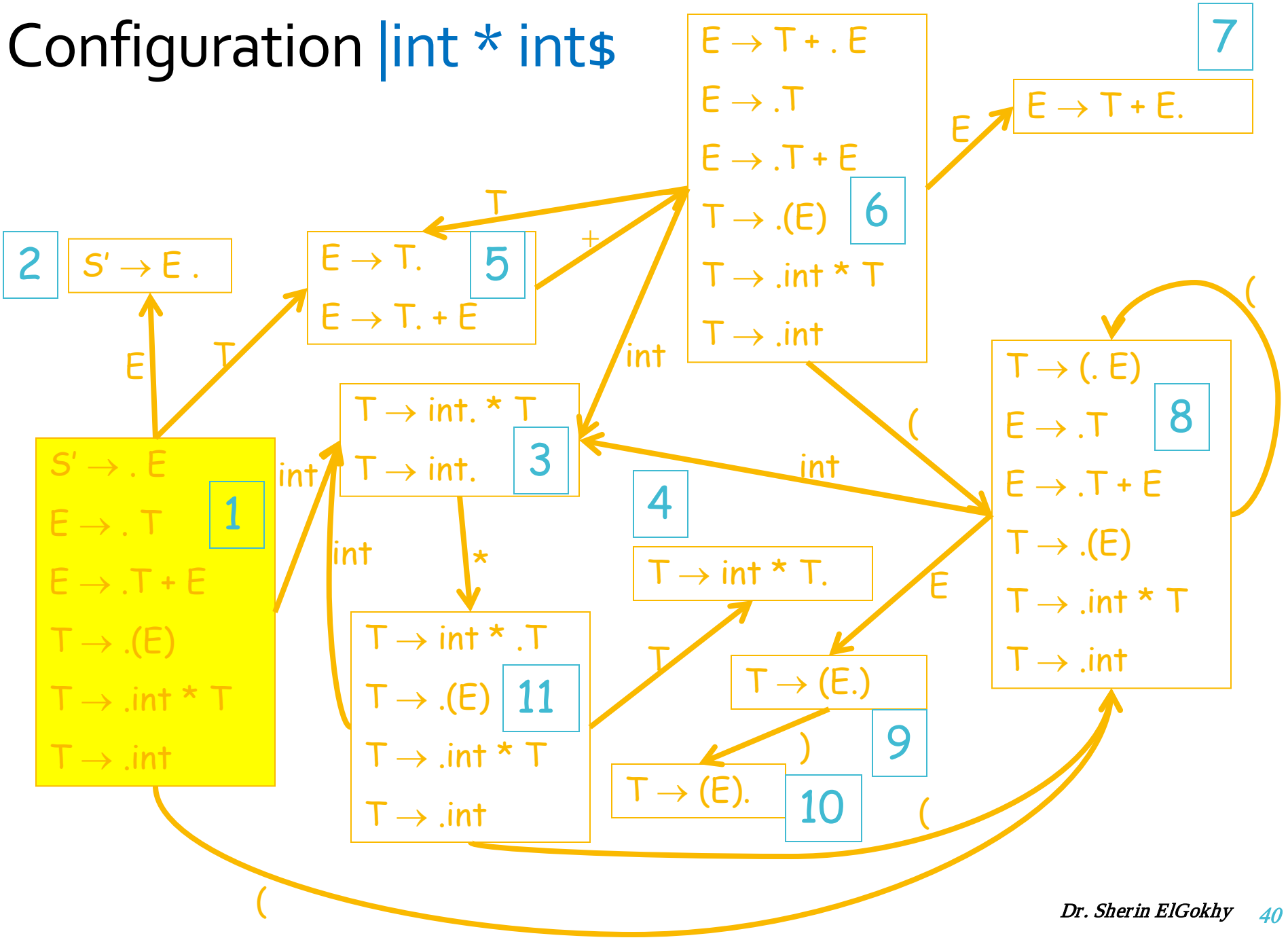
Notes

- If there is a conflict in the last step, grammar is not SLR(k)
- k is the amount of lookahead
  - In practice k = 1

SLR Example

Configuration DFA	*Halt State*	*Action*
|int * int$	1	shift

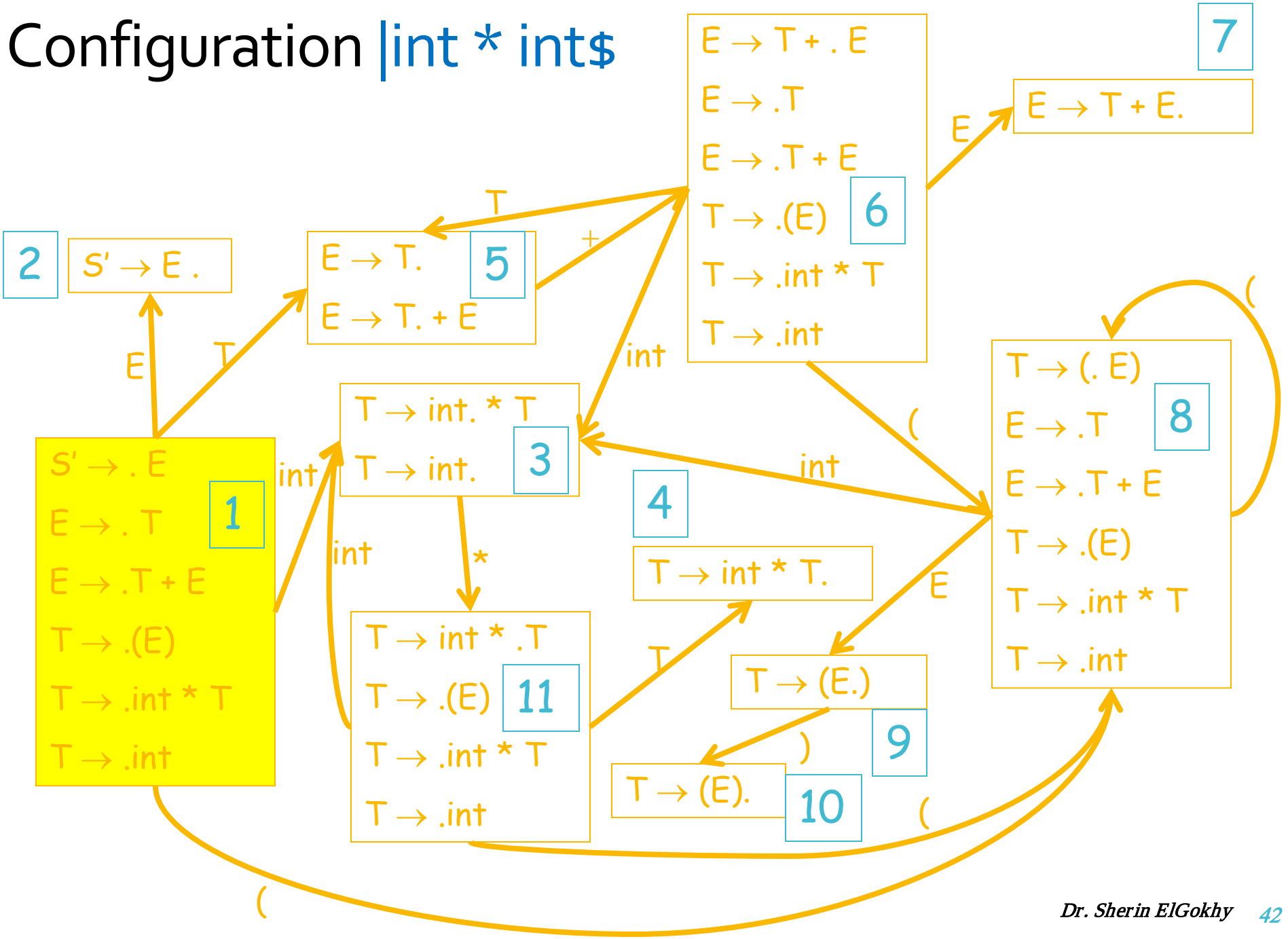
Configuration |int * int$



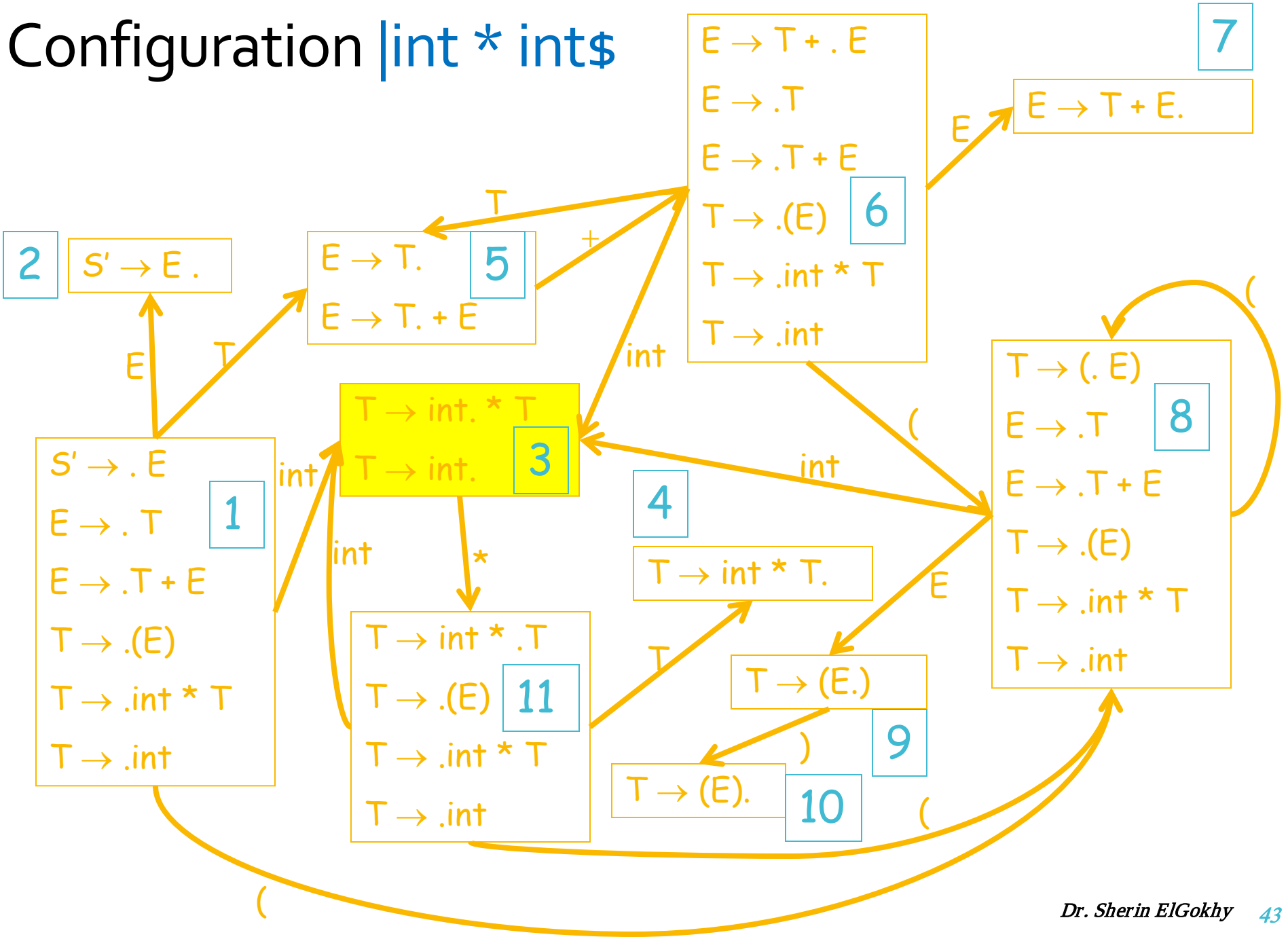
SLR Example

Configuration DFA	*Halt State*		*Action*
|int * int$	1		shift
int | * int$	3	* not in Follow(T)	shift

Configuration |int * int$



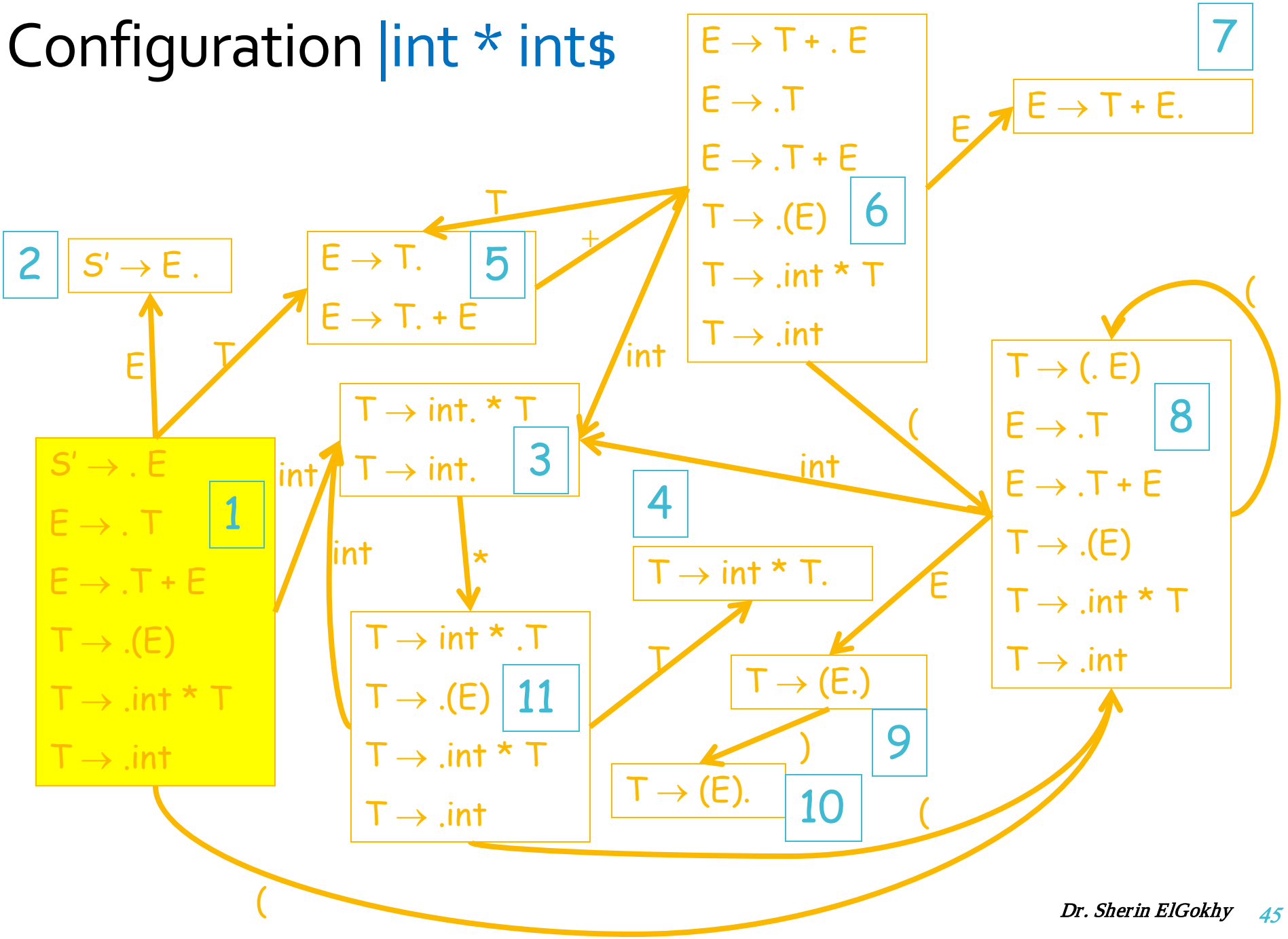
Configuration |int * int$



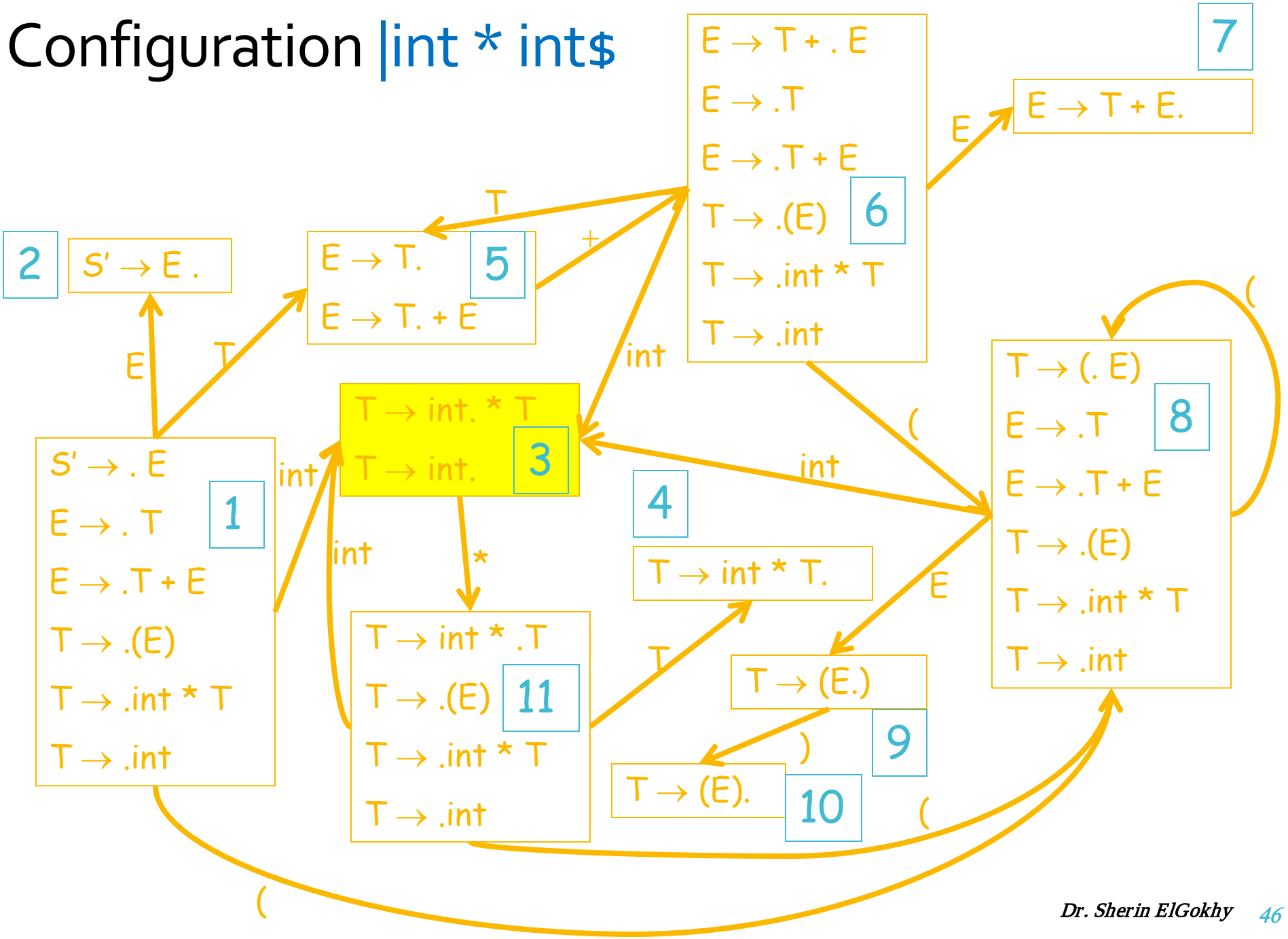
SLR Example

Configuration	*DFA Halt State*	*Action*
|int * int$	1	shift
int | * int$	3 * not in Follow(T)	shift
int * | int$	11	shift

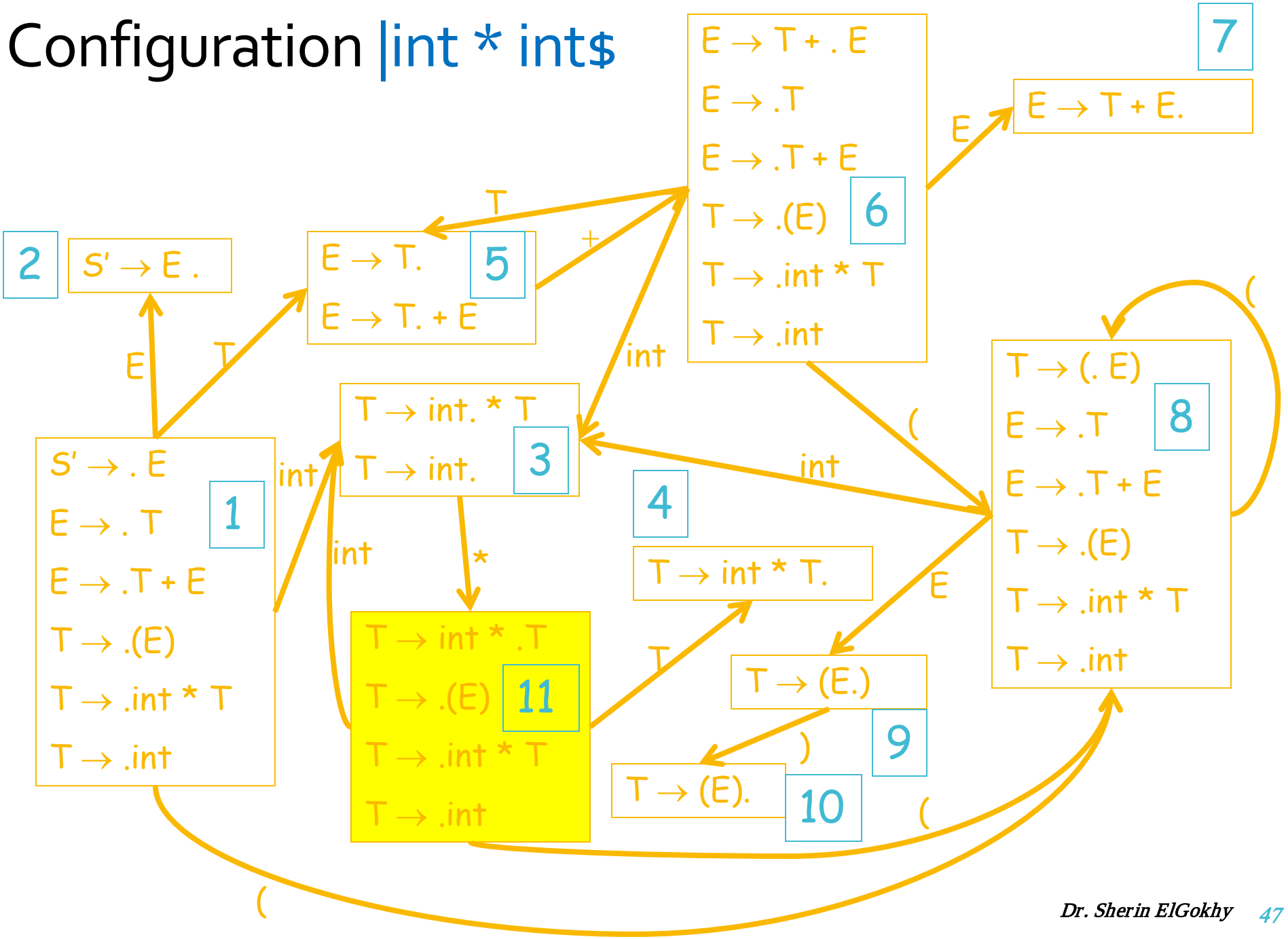
Configuration |int * int$



Configuration |int * int$



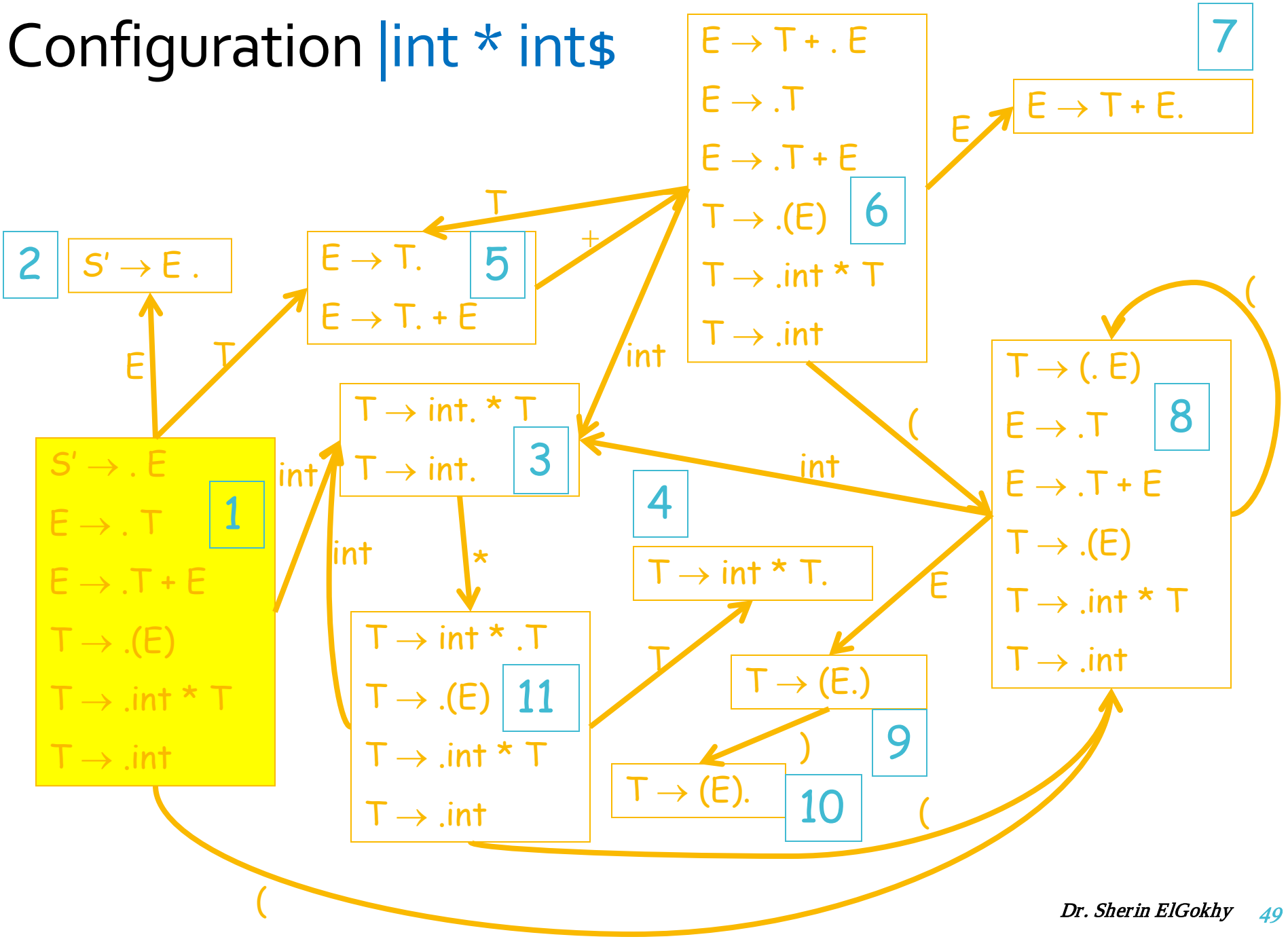
Configuration |int * int$



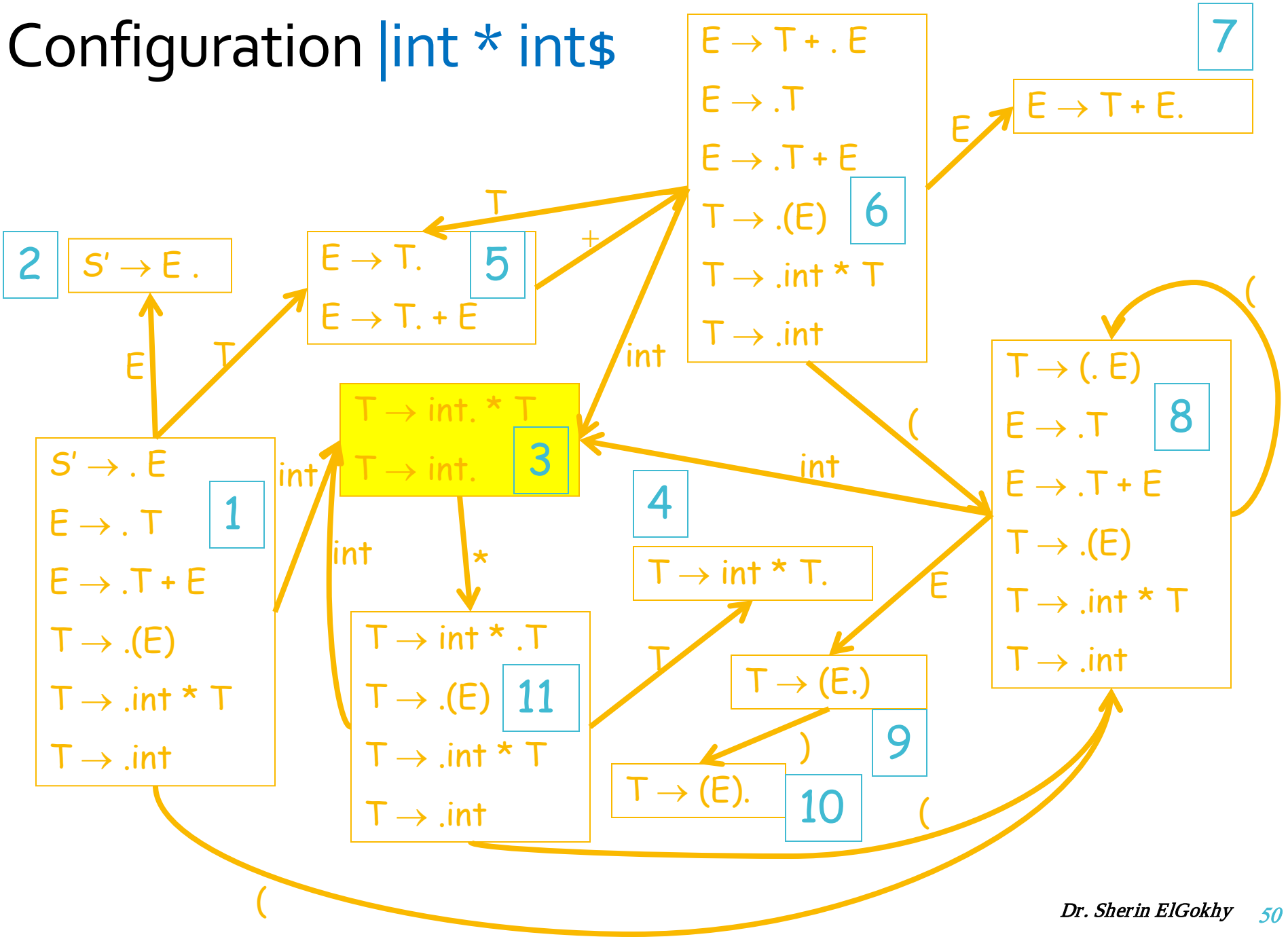
SLR Example

Configuration	*DFA Halt State*	*Action*
|int * int$	1	shift
int | * int$	3 * not in Follow(T)	shift
int * | int$	11	shift
int * int |$	3 $ ∈ Follow(T)	red. T→int

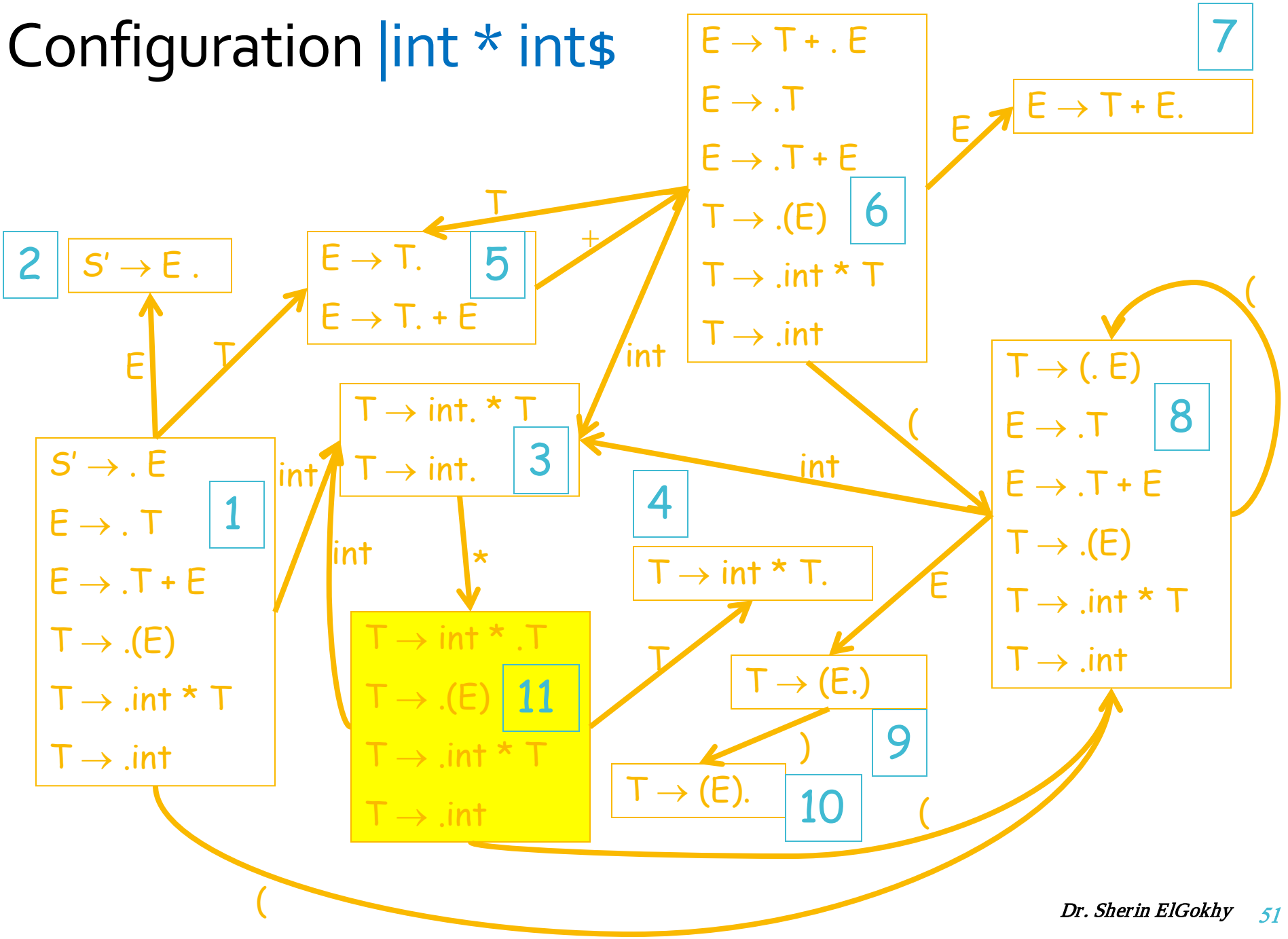
Configuration |int * int$



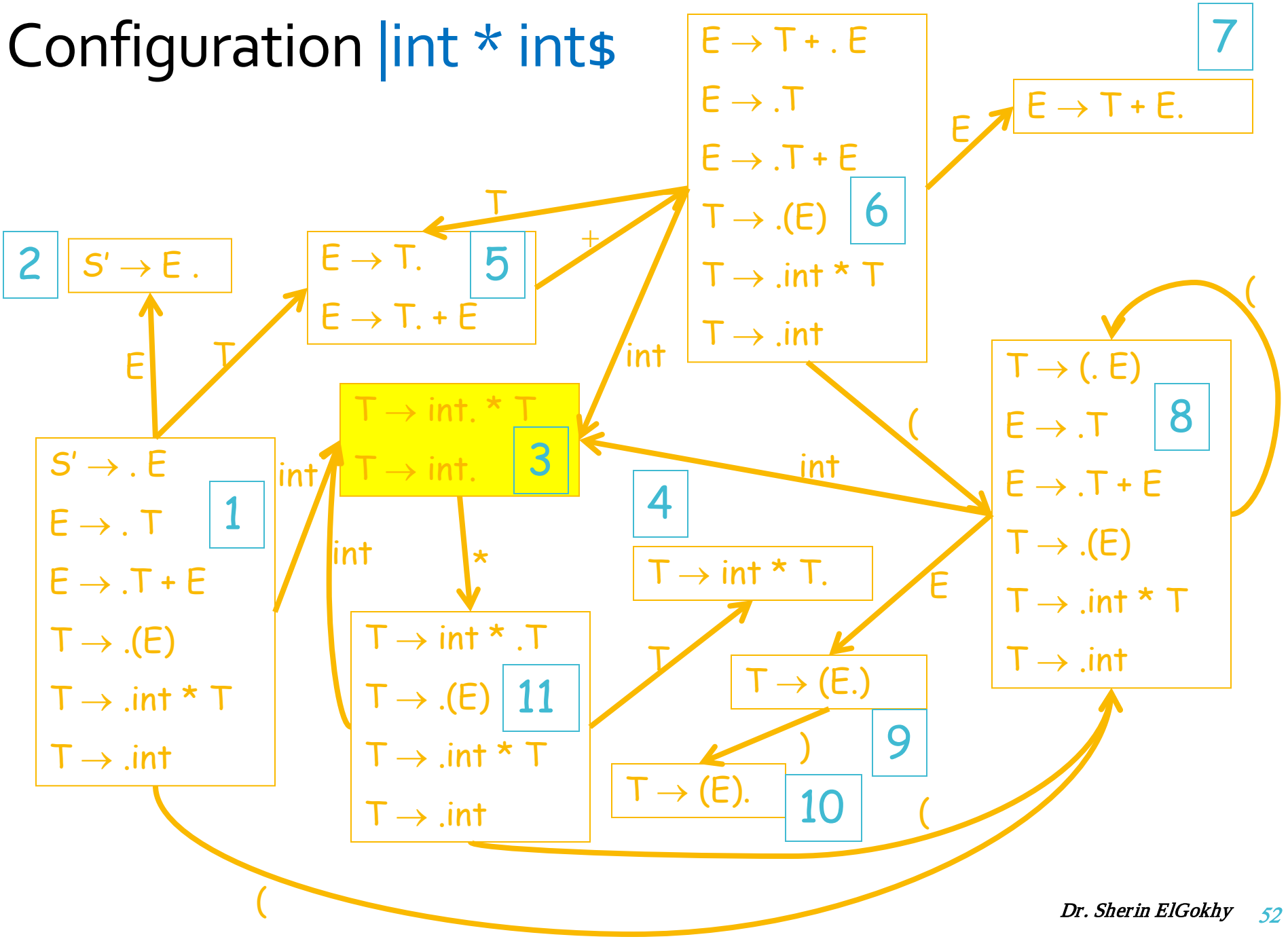
Configuration |int * int$



Configuration |int * int$



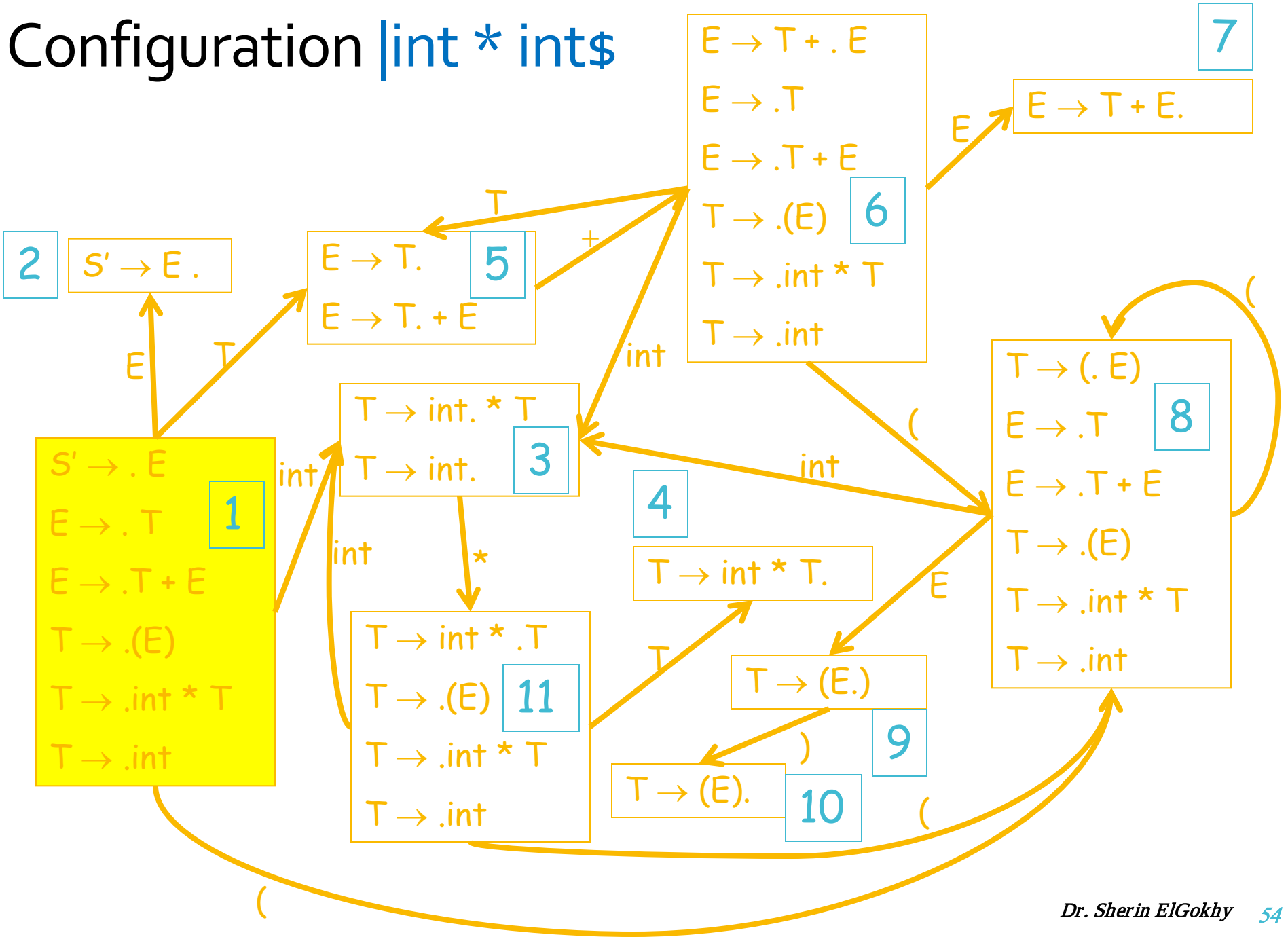
Configuration |int * int$



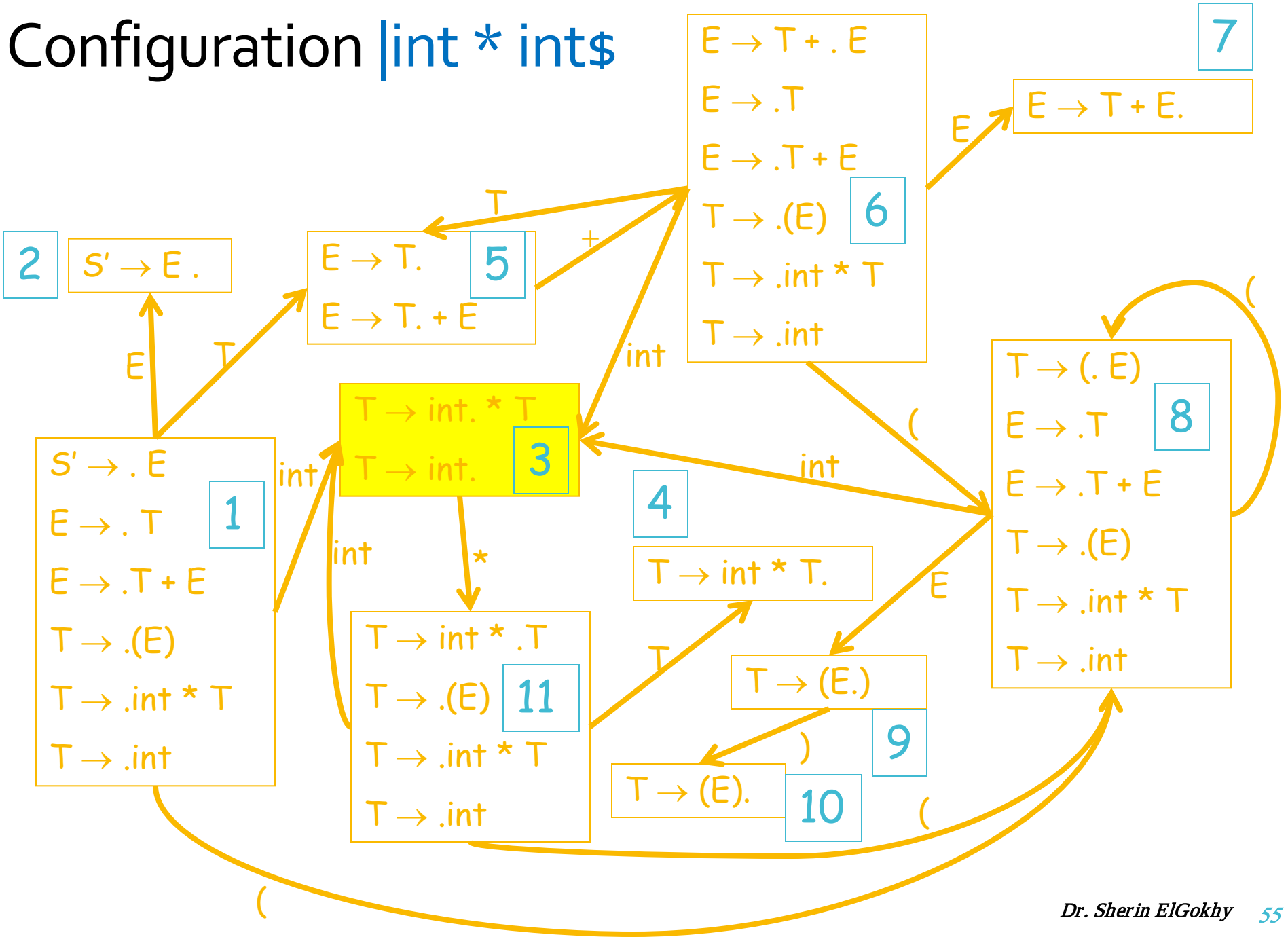
SLR Example

Configuration	*DFA Halt State*	*Action*
|int * int$	1	shift
int | * int$	3 * not in Follow(T)	shift
int * | int$	11	shift
int * int |$	3 $ ∈ Follow(T)	red. T→int
int * T |$	4 $ ∈ Follow(T)	red. T→int*T

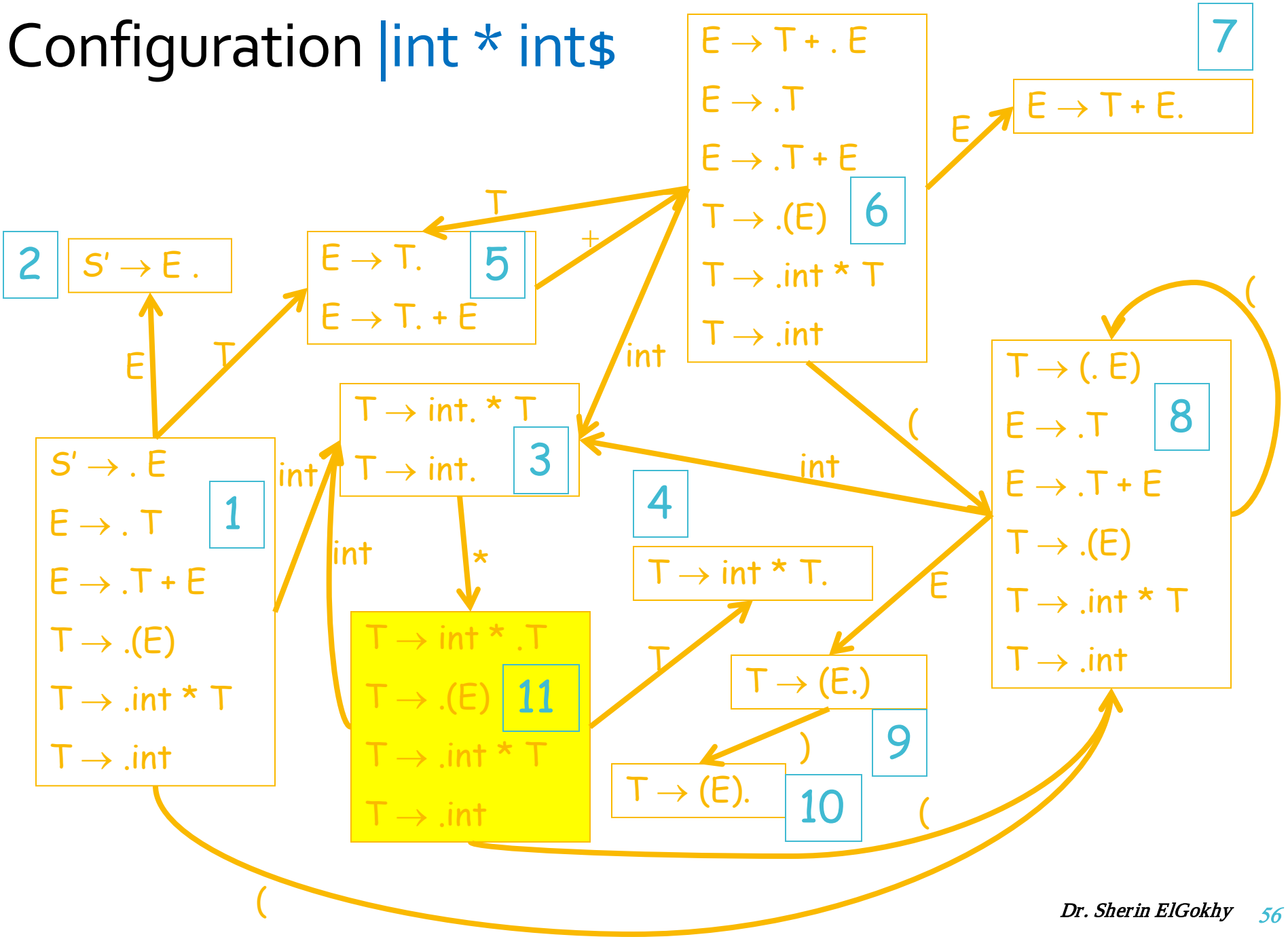
Configuration |int * int$



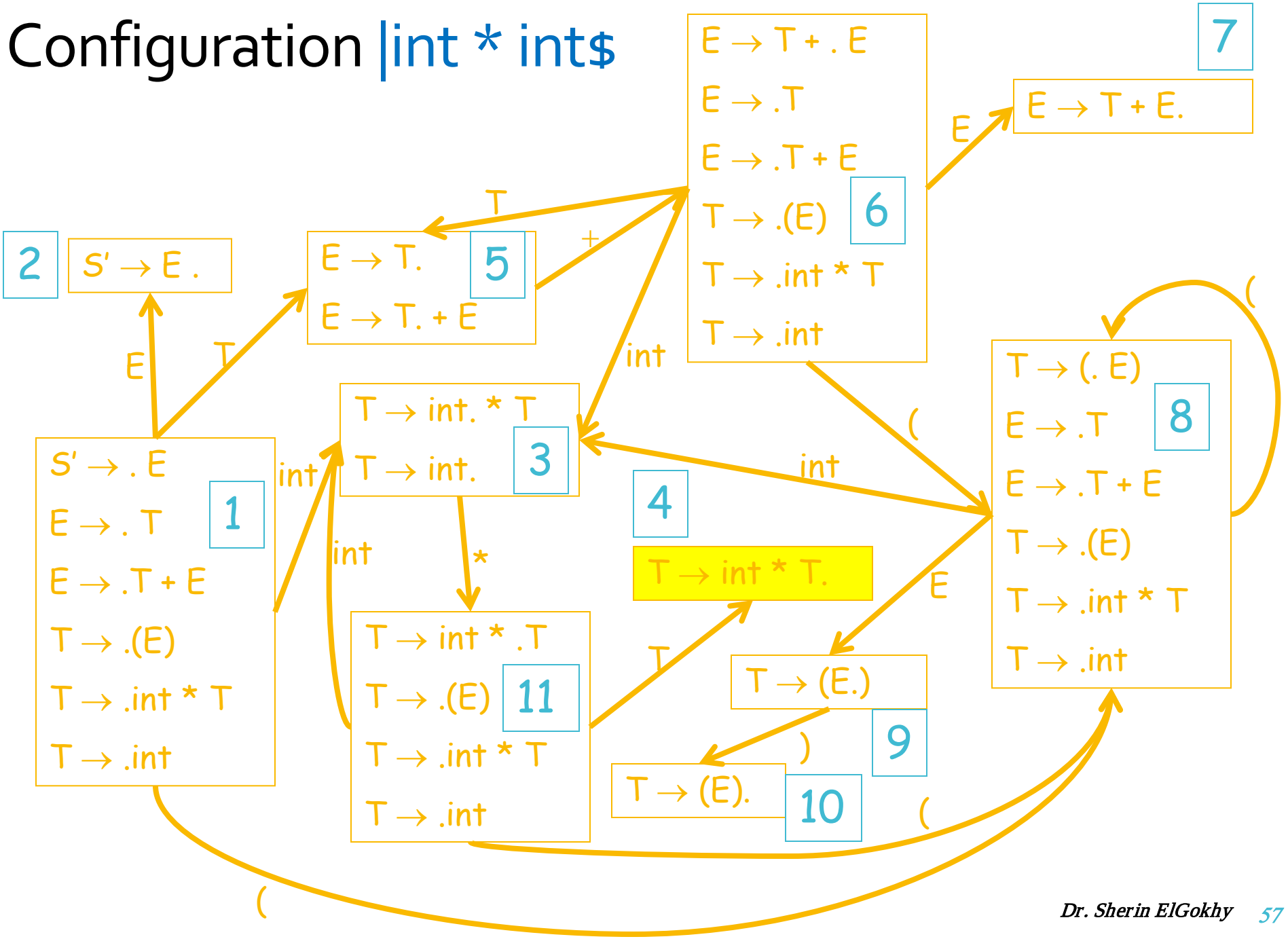
Configuration |int * int$



Configuration |int * int$



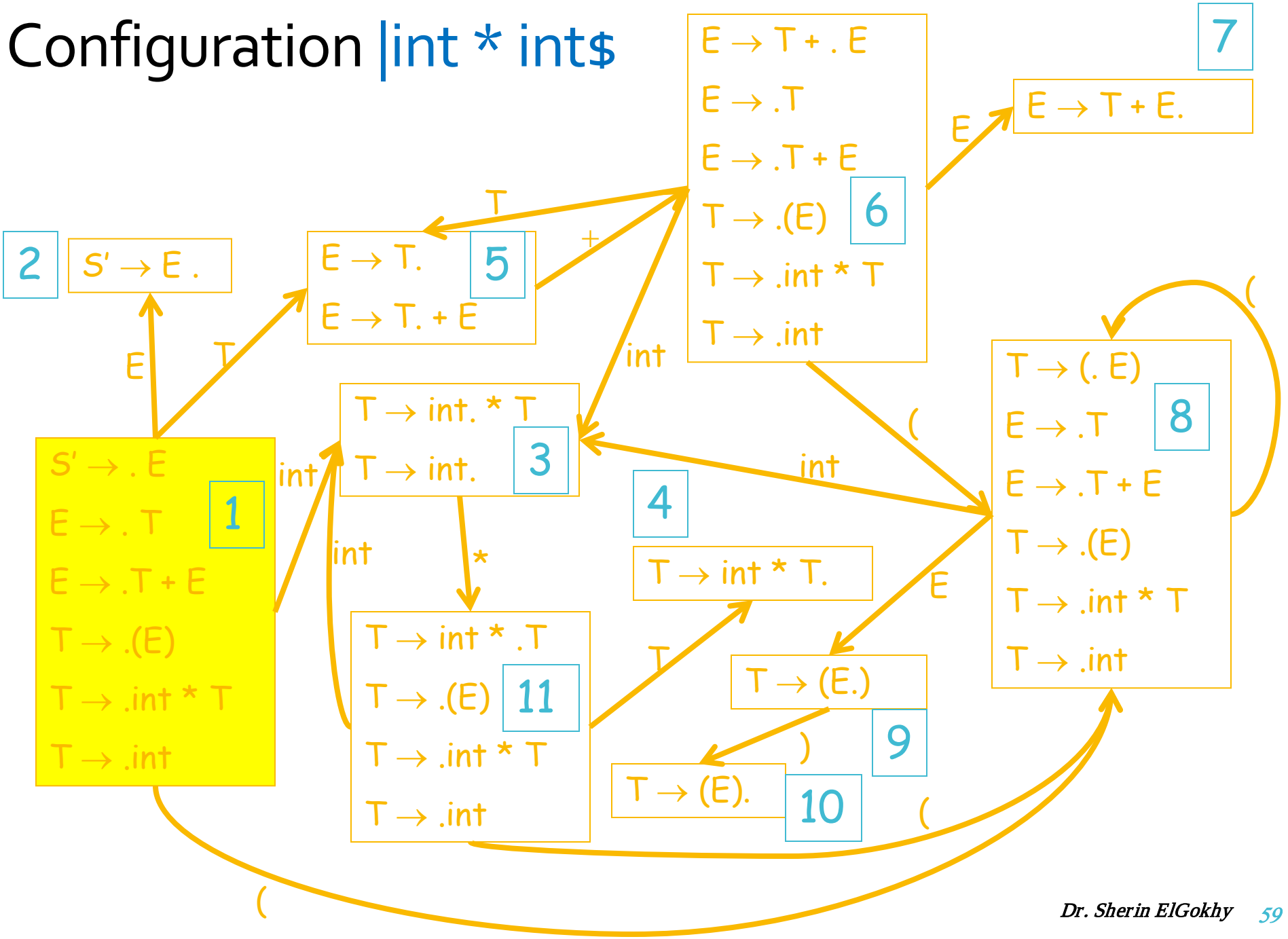
Configuration |int * int$



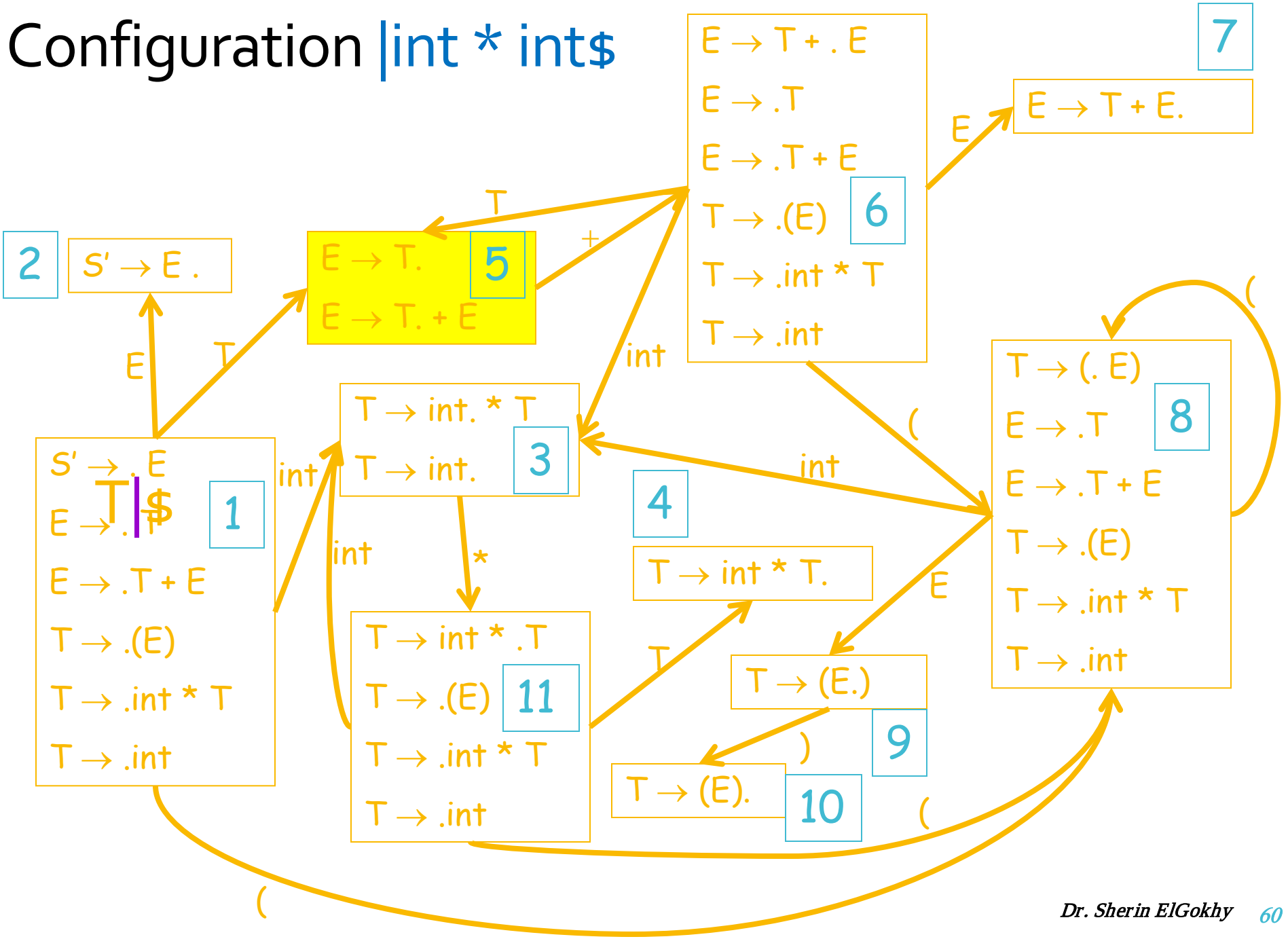
SLR Example

Configuration	*DFA Halt State*	*Action*
|int * int$	1	shift
int | * int$	3 * not in Follow(T)	shift
int * | int$	11	shift
int * int |$	3 $ ∈ Follow(T)	red. T→int
int * T |$	4 $ ∈ Follow(T)	red. T→int*T
T |$	5 $ ∈ Follow(E)	red. E→T

Configuration |int * int$



Configuration |int * int$



SLR Example

Configuration	*DFA Halt State*	*Action*
|int * int$	1	shift
int | * int$	3 * not in Follow(T)	shift
int * | int$	11	shift
int * int |$	3 $ ∈ Follow(T)	red. T→int
int * T |$	4 $ ∈ Follow(T)	red. T→int*T
T |$	5 $ ∈ Follow(E)	red. E→T
E |$		accept

Quiz

Using the DFA on the previous slide, choose the next action for the given parse state

Configuration	*DFA Halt State*
int * int|+ int $	3

- ○ shift
- ○ red. T → int
- ○ red. T → int * T
- ○ accept

Thanks